The Project Gutenberg eBook of Hackers, Heroes of the Computer Revolution. Chapters 1 and 2, by Steven Levy

This is a *copyrighted* Project Gutenberg eBook, details below.

Title: Hackers, Heroes of the Computer Revolution. Chapters 1 and 2

Author: Steven Levy

Release date: November 1, 1996 [EBook #729]
Most recently updated: April 11, 2013

Language: English

*** START OF THE PROJECT GUTENBERG EBOOK HACKERS, HEROES OF THE COMPUTER REVOLUTION. CHAPTERS 1 AND 2 ***

Hackers, Heroes of the Computer Revolution, by Steven Levy

(C)1984 by Steven Levy

Chapters 1 and 2 of Hackers, Heroes of the Computer Revolution by Steven Levy

Who's Who
The Wizards and their Machines

Bob Albrecht Found of People's Computer Company who took visceral pleasure in exposing youngsters to computers.

Altair 8800
The pioneering microcomputer that galvanized hardware hackers.
Building this kit made you learn hacking. Then you tried to
figure out what to DO with it.

Apple II][Steve Wozniak's friendly, flaky, good-looking computer, wildly successful and the spark and soul of a thriving industry.

Atari 800 This home computer gave great graphics to game hackers like John Harris, though the company that made it was loath to tell you how it worked.

Bob and Carolyn Box World-record-holding gold prospectors turned software stars, working for Sierra On-Line.

Doug Carlston Corporate lawyer who chucked it all to form the Broderbund software company.

Bob Davis
Left job in liquor store to become best-selling author
of Sierra On-Line computer game "Ulysses and the Golden Fleece."
Success was his downfall.

Peter Deutsch Bad in sports, brilliant at math, Peter was still in short pants when he stubled on the TX-0 at MIT—and hacked it along with the masters.

Steve Dompier Homebrew member who first made the Altair sing, and later wrote the "Targe" game on the Sol which entranced Tom Snyder.

John Draper
The notorious "Captain Crunch" who fearlessly explored
the phone systems, got jailed, hacked microprocessors.
Cigarettes made his violent.

Mark Duchaineau The young Dungeonmaster who copy-protected On-Lines disks at his whim.

Chris Esponosa Fourteen-year-old follower of Steve Wozniak and early Apple employee.

Lee Felsenstein Former "military editor" of Berkeley Barb, and hero of an imaginary science-fiction novel, he designed computers with "junkyard" approach and was central figure in Bay Area hardware hacking in the seventies.

Ed Fredkin Gentle founder of Information International, thought himself world's greates programmer until he met Stew Nelson. Father figure to hackers.

Gordon French Silver-haired hardware hacker whose garage held not cars but his homebrewed Chicken Hawk comptuer, then held the first Homebrew Computer Club meeting.

Richard Garriott Astronaut's son who, as Lord British, created Ultima world on computer disks.

Bill Gates Cocky wizard, Harvard dropout who wrote Altair BASIC, and complained when hackers copied it.

Bill Gosper
Horwitz of computer keyboards, master math and LIFE hacker
at MIT AI lab, guru of the Hacker Ethic and student of
Chinese restaurant menus.

Richard Greenblatt Single-minded, unkempt, prolific, and canonical MIT hacker who went into night phase so often that he zorched his academic career. The hacker's hacker.

John Harris The young Atari 800 game hacker who became Sierra On-Line's star programmer, but yearned for female companionship.

IBM-PC IBM's entry into the personal computer market which amazingly included a bit of the Hacker Ethic, and took over. [H.E. as open architecture.]

IBM 704
IBM was The Enemy, and this was its machine,
the Hulking Giant computer in MIT's Building 26.
Later modified into the IBM 709, then the IBM 7090.
Batch-processed and intolerable.

Jerry Jewell
Vietnam vet turned programmer who founded Sirius Software.

Steven Jobs
Visionary, beaded, non-hacking youngster who took
Wozniak's Apple II][, made a lot of deals,
and formed a company that would make a billion dollars.

Tom Knight
At sixteen, an MIT hacker who would name the
Incompatible Time-sharing System. Later a
Greenblatt nemesis over the LISP machine schism.

Alan Kotok The chubby MIT student from Jersey who worked under the rail layout at TMRC, learned the phone system at Western Electric, and became a legendary TX-0 and PDP-1 hacker.

Effrem Lipkin Hacker-activist from New York who loved machines but hated their uses. Co-Founded Community Memory; friend of Felsenstein.

LISP Machine The ultimate hacker computer, invented mosly by Greenblatt and subject of a bitter

dispute at MIT.

"Uncle" John McCarthy Absent-minded but brilliant MIT [later Stanford] professor who helped pioneer computer chess, artificial intelligence, LISP.

Bob Marsh Berkeley-ite and Homebrewer who shared garage with Felsenstein and founded Processor Technology, which made the Sol computer.

Roger Melen Homebrewer who co-founded Cromemco company to make circuit boards for Altair. His "Dazzler" played LIFE programs on his kitchen table.

Louis Merton Pseudonym for the AI chess hacker whose tendency to go catatonic brought the hacker community together.

Jude Milhon
Met Lee Felsenstein through a classified ad in the
Berkeley Barb, and became more than a friend—
a member of the Community Memory collective.

Marvin Minsky Playful and brilliant MIT prof who headed the AI lave and allowed the hackers to run free.

Fred Moore Vagabond pacifist who hated money, loved technology, and co-founded Homebrew Club.

Stewart Nelson
Buck-toothed, diminutive, but fiery AI lab hacker
who connected the PDP-1 comptuer to hack the phone system.
Later co-founded the Systems Concepts company.

Ted Nelson Self-described "innovator" and noted curmudgeon who self-published the influential Computer Lib book.

Russel Noftsker Harried administrator of MIT AI lab in the late sixties; later president of Symbolics company.

Adam Osborne
Bangkok-born publisher-turned-computer-manufacturer
who considered himself a philsopher. Founded Osborne
Computer Company to make "adequate" machines.

PDP-1 Digital Equipment's first minicomputer, and in 1961 an interactive godsend to the MIT hackers and a slap in the face to IBM fascism.

PDP-6 Designed in part by Kotok, this mainframe computer was cornerstone of AI lab, with its gorgeious instruction set and sixteen sexy registers.

Tom Pittman The religious Homebrew hacker who lost his wife but kept the faith with his Tiny Basic.

Ed Roberts Enigmatic founder of MITS company who shook the world with his Altair computer. He wanted to help people build mental pyramids.

Steve [Slug] Russell McCarthy's "coolie," who hacked the Spacewar program, first videogame, on the PDP-1. Never made a dime from it.

Peter Samson
MIT hacker, one of the first, who loved systems, trains,
TX-0, music, parliamentary procedure, pranks, and hacking.

Bob Saunders Jolly, balding TMRC hacker who married early, hacked till late at night eating "lemon gunkies," and mastered the "CBS Strategy on Spacewar.

Warren Schwader Big blond hacker from rural Wisconsin who went from the assembly line to software stardom but couldn't reconcile the shift with his devotion to Jehovah's Witnesses.

David Silver Left school at fourteen to be mascot of AI lab; maker of illicit keys and builder of a tiny robot that did the impossible.

Dan Sokol Long-haired prankster who reveled in revealing technological secrets at Homebrew Club. Helped "liberate" Alair BASIC on paper tape.

Les Solomon Editor of Popular Electroics, the puller of strings who set the computer revolution into motion.

Marty Spergel The Junk Man, the Homebrew member who supplied circuits and cables and could make you a deal for anything.

Richard Stallman
The Last of the Hackers, who vowed to defend
the principles of Hackerism to the bitter end.
Remained at MIT until there was no one to eat
Chinese food with.

Jeff Stephenson Thirty-year-old martial arts veteran and hacker who was astounded that joining Sierra On-Line meant enrolling in Summer Camp.

Jay Sullivan MAddeningly clam wizard-level programmer at Informatics who impressed Ken Williams by knowing the meaning of the word "any."

Dick Sunderland Chalk-complexioned MBA who believed that firm managerial bureaucracy was a worth goal, but as president of Sierra On-Line found that hackers didn't think that way.

Gerry Sussman Young MIT hacker branded "loser" because he smoked a pipe and "munged" his programs; later became "winner" by algorithmic magic.

Margot Tommervik With her husband Al, long-haired Margot parlayed her game show winnings into a magazine that deified the Apple Computer.

Tom Swift Terminal Lee Felsenstein's legendary, never-to-be-built computer terminal which would give the user ultimate leave to get his hands on the world.

TX-0
Filled a small room, but in the late fifties this $3 million machine
was the world's first personal computer—for the community of
MIT hackers that formed around it.

Jim Warren Portly purveyor of "techno-gossip" at Homebrew, he was first editor of hippie-styled Dr. Dobbs Journal, later started the lucrative Computer Faire.

Randy Wigginton
Fifteen-year-old member of Steve Wozniak's kiddie corps,
he help Woz trundle the Apple II to Homebrew.
Still in high school when he became Apple's first software employee.

Ken Williams Arrogant and brilliant young programmer who saw the writing on the CRT and started Sierra On-Line to make a killing and improve society by selling games for the Apple computer.

Roberta Williams Ken Williams' timid wife who rediscovered her own creativity by writing "Mystery House," the first of her many bestselling computer games.

Steven "Woz" Wozniak Openhearted, technologically daring hardware hacker from San Jose suburbs. Woz built the Apple Computer for the pleasure of himself and friends.

PART ONE True Hackers CAMBRIDGE: The Fifties and Sixties

CHAPTER 1 THE TECH MODEL RAILROAD CLUB

Just why Peter Samson was wandering around in Building 26 in the middle of the night is a matter that he would find difficult to explain. Some things are not spoken. If you were like the people whom Peter Samson was coming to know and befriend in this, his freshman year at the Massachusetts Institute of Technology in the winter of 1958-59, no explanation would be required. Wandering around

the labyrinth of laboratories and storerooms, searching for the secrets of telephone switching in machine rooms, tracing paths of wires or relays in subterranean steam tunnels . . . for some, it was common behavior, and there was no need to justify the impulse, when confronted with a closed door with an unbearably intriguing noise behind it, to open the door uninvited. And then, if there was no one to physically bar access to whatever was making that intriguing noise, to touch the machine, start flicking switches and noting responses, and eventually to loosen a screw, unhook a template, jiggle some diodes and tweak a few connections. Peter Samson and his friends had grown up with a specific relationship to the world, wherein things had meaning only if you found out how they worked. And how would you go about that if not by getting your hands on them?

It was in the basement of Building 26 that Samson and his friends discovered the EAM room. Building 26 was a long glass-and-steel structure, one of MIT's newer buildings, contrasting with the venerable pillared structures that fronted the Institute on Massachusetts Avenue. In the basement of this building void of personality, the EAM room. Electronic Accounting Machinery. A room that housed machines which ran like computers.

Not many people in 1959 had even seen a computer, let alone touched one. Samson, a wiry, curly-haired redhead with a way of extending his vowels so that it would seem he was racing through lists of possible meanings of statements in mid-word, had viewed computers on his visits to MIT from his hometown of Lowell, Massachusetts, less than thirty miles from campus. This made him a "Cambridge urchin," one of dozens of science-crazy high schoolers in the region who were drawn, as if by gravitational pull, to the Cambridge campus. He had even tried to rig up his own computer with discarded parts of old pinball machines: they were the best source of logic elements he could find.

LOGIC ELEMENTS: the term seems to encapsulate what drew Peter Samson, son of a mill machinery repairman, to electronics. The subject made sense. When you grow up with an insatiable curiosity as to how things work, the delight you find upon discovering something as elegant as circuit logic, where all connections have to complete their loops, is profoundly thrilling. Peter Samson, who early on appreciated the mathematical simplicity of these things, could recall seeing a television show on Boston's public TV channel, WGBH, which gave a rudimentary introduction to programming a computer in its own language. It fired his imagination: to Peter Samson, a computer was surely like Aladdin's lamp—rub it, and it would do your bidding. So he tried to learn more about the field, built machines of his own, entered science project competitions and contests, and went to the place that people of his ilk aspired to: MIT. The repository of the very brightest of those weird high school kids with owl-like glasses and underdeveloped pectorals who dazzled math teachers and flunked PE, who dreamed not of scoring on prom night, but of getting to the finals of the General Electric Science Fair competition. MIT, where he would wander the hallways at two o'clock in the morning, looking for something interesting, and where he would indeed discover something that would help draw him deeply into a new form of creative process, and a new life-style, and would put him into the forefront of a society envisioned only by a few science-fiction writers of mild disrepute. He would discover a computer that he could play with.

The EAM room which Samson had chanced on was loaded with large keypunch machines the size of squat file cabinets. No one was protecting them: the room was staffed only by day, when a select group who had attained official clearance were privileged enough to submit long manila cards to operators who would then use these machines to punch holes in them according to what data the privileged ones wanted entered on the cards. A hole in the card would represent some instruction to the computer, telling it to put a piece of data somewhere, or perform a function on a piece of data, or move a piece of data from one place to another. An entire stack of these cards made one computer program, a program being a series of instructions which yield some expected result, just as the instructions in a recipe, when precisely followed, lead to a cake. Those cards would be taken to yet another operator upstairs who would feed the cards into a "reader" that would note where the holes were and dispatch this information to the IBM 704 computer on the first floor of Building 26. The Hulking Giant.

The IBM 704 cost several million dollars, took up an entire room, needed constant attention from a cadre of professional machine operators, and required special air-conditioning so that the glowing vacuum tubes inside it would not heat up to data-destroying temperatures. When the air-conditioning broke down—a fairly common occurrences—a loud gong would sound, and three engineers would spring from a nearby office to frantically take covers off the machine so its innards wouldn't melt. All these people in charge of punching cards, feeding them into readers, and pressing buttons and switches on the machine were what was commonly called a Priesthood, and those privileged enough to submit data to those most holy priests were the official acolytes. It was an almost ritualistic exchange.

ACOLYTE: Oh machine, would you accept my offer of information so you may run my program and perhaps give me a computation?

PRIEST (on behalf of the machine): We will try. We promise nothing.

As a general rule, even these most privileged of acolytes were not allowed direct access to the machine itself, and they would not be able to see for hours, sometimes for days, the results of the machine's ingestion of their "batch" of cards.

This was something Samson knew, and of course it frustrated the hell out of Samson, who wanted to get at the damn machine. For this was what life was all about.

What Samson did not know, and was delighted to discover, was that the EAM room also had a particular keypunch machine called the 407. Not only could it punch cards, but it could also read cards, sort them, and print them on listings. No one seemed to be guarding these machines, which were computers, sort of. Of course, using them would be no picnic: one needed to actually wire up what was called a plug board, a two-inch-by-two-inch plastic square with a mass of holes in it. If you put hundreds of wires through the holes in a certain order, you would get something that looked like a rat's nest but would fit into this electromechanical machine and alter its personality. It could do what you wanted it to do.

So, without any authorization whatsoever, that is what Peter Samson set out to do, along with a few friends of his from an MIT organization with a special interest in model railroading. It was a casual, unthinking step into a science-fiction future, but that was typical of the way that an odd subculture was pulling itself up by its bootstraps and growing to underground prominence—to become a culture that would be the impolite, unsanctioned soul of computerdom. It was among the first computer hacker escapades of the Tech Model Railroad Club, or TMRC.

* * *

Peter Samson had been a member of the Tech Model Railroad Club since his first week at MIT in the fall of 1958. The first event that entering MIT freshmen attended was a traditional welcoming lecture, the same one that had been given for as long as anyone at MIT could remember. LOOK AT THE PERSON TO YOUR LEFT . . . LOOK AT THE PERSON TO YOUR RIGHT . . . ONE OF YOU THREE WILL NOT GRADUATE FROM THE INSTITUTE. The intended effect of the speech was to create that horrid feeling in the back of the collective freshman throat that signaled unprecedented dread. All their lives, these freshmen had been almost exempt from academic pressure. The exemption had been earned by virtue of brilliance. Now each of them had a person to the right and a person to the left who was just as smart. Maybe even smarter.

But to certain students this was no challenge at all. To these youngsters, classmates were perceived in a sort of friendly haze: maybe they would be of assistance in the consuming quest to find out how things worked, and then to master them. There were enough obstacles to learning already—why bother with stupid things like brown-nosing teachers and striving for grades? To students like Peter Samson, the quest meant more than the degree.

Sometime after the lecture came Freshman Midway. All the campus organizations—special-interest groups, fraternities, and such— set up booths in a large gymnasium to try to recruit new members. The group that snagged Peter was the Tech Model Railroad Club. Its members, bright-eyed and crew-cutted upperclassmen who spoke with the spasmodic cadences of people who want words out of the way in a hurry, boasted a spectacular display of HO gauge trains they had in a permanent clubroom in Building 20. Peter Samson had long been fascinated by trains, especially subways. So he went along on the walking tour to the building, a shingle-clad temporary structure built during World War II. The hallways were cavernous, and even though the clubroom was on the second floor it had the dank, dimly lit feel of a basement.

The clubroom was dominated by the huge train layout. It just about filled the room, and if you stood in the little control area called "the notch" you could see a little town, a little industrial area, a tiny working trolley line, a papier-mache mountain, and of course a lot of trains and tracks. The trains were meticulously crafted to resemble their full-scale counterparts, and they chugged along the twists and turns of track with picture-book perfection.

And then Peter Samson looked underneath the chest-high boards which held the layout. It took his breath away. Underneath this layout was a more massive matrix of wires and relays,and crossbar switches than Peter Samson had ever dreamed existed. There were neat regimental lines of switches, and achingly regular rows of dull bronze relays, and a long, rambling tangle of red, blue, and yellow wires—twisting and twirling like a rainbow-colored explosion of Einstein's hair. It was an incredibly complicated system, and Peter Samson vowed to find out how it worked.

The Tech Model Railroad Club awarded its members a key to the clubroom after they logged forty

hours of work on the layout. Freshman Midway had been on a Friday. By Monday, Peter Samson had his key.

* * *

There were two factions of TMRC. Some members loved the idea of spending their time building and painting replicas of certain trains with historical and emotional value, or creating realistic scenery for the layout. This was the knife-and-paintbrush contingent, and it subscribed to railroad magazines and booked the club for trips on aging train lines. The other faction centered on the Signals and Power Subcommittee of the club, and it cared far more about what went on under the layout. This was The System, which worked something like a collaboration between Rube Goldberg and Wernher von Braun, and it was constantly being improved, revamped, perfected, and sometimes "gronked"—in club jargon, screwed up. S&P people were obsessed with the way The System worked, its increasing complexities, how any change you made would affect other parts, and how you could put those relationships between the parts to optimal use.

Many of the parts for The System had been donated by the Western Electric College Gift Plan, directly from the phone company. The club's faculty advisor was also in charge of the campus phone system, and had seen to it that sophisticated phone equipment was available for the model railroaders. Using that equipment as a starting point, the Railroaders had devised a scheme which enabled several people to control trains at once, even if the trains were at different parts of the same track. Using dials appropriated from telephones, the TMRC "engineers" could specify which block of track they wanted control of, and run a train from there. This was done by using several types of phone company relays, including crossbar executors and step switches which let you actually hear the power being transferred from one block to another by an other-worldly chunka-chunka-chunka sound.

It was the S&P group who devised this fiendishly ingenious scheme, and it was the S&P group who harbored the kind of restless curiosity which led them to root around campus buildings in search of ways to get their hands on computers. They were lifelong disciples of a Hands-On Imperative. Head of S&P was an upperclassman named Bob Saunders, with ruddy, bulbous features, an infectious laugh, and a talent for switch gear. As a child in Chicago, he had built a high-frequency transformer for a high school project; it was his six-foot-high version of a Tesla coil, something devised by an engineer in the 1800s which was supposed to send out furious waves of electrical power. Saunders said his coil project managed to blow out television reception for blocks around. Another person who gravitated to S&P was Alan Kotok, a plump, chinless, thick-spectacled New Jerseyite in Samson's class. Kotok's family could recall him, at age three, prying a plug out of a wall with a screwdriver and causing a hissing shower of sparks to erupt. When he was six, he was building and wiring lamps. In high school he had once gone on a tour of the Mobil Research Lab in nearby Haddonfield, and saw his first computer—the exhilaration of that experience helped him decide to enter MIT. In his freshman year, he earned a reputation as one of TMRC's most capable S&P people.

The S&P people were the ones who spent Saturdays going to Eli Heffron's junkyard in Somerville scrounging for parts, who would spend hours on their backs resting on little rolling chairs they called "bunkies" to get underneath tight spots in the switching system, who would work through the night making the wholly unauthorized connection between the TMRC phone and the East Campus. Technology was their playground.

The core members hung out at the club for hours; constantly improving The System, arguing about what could be done next, developing a jargon of their own that seemed incomprehensible to outsiders who might chance on these teen-aged fanatics, with their checked short-sleeve shirts, pencils in their pockets, chino pants, and, always, a bottle of Coca-Cola by their side. (TMRC purchased its own Coke machine for the then forbidding sum of $165; at a tariff of five cents a bottle, the outlay was replaced in three months; to facilitate sales, Saunders built a change machine for Coke buyers that was still in use a decade later.) When a piece of equipment wasn't working, it was "losing"; when a piece of equipment was ruined, it was "munged" (Mash Until No Good); the two desks in the corner of the room were not called the office, but the "orifice"; one who insisted on studying for courses was a "tool"; garbage was called "cruft"; and a project undertaken or a product built not solely to fulfill some constructive goal, but with some wild pleasure taken in mere involvement, was called a "hack."

This latter term may have been suggested by ancient MIT lingo— the word "hack" had long been used to describe the elaborate college pranks that MIT students would regularly devise, such as covering the dome that overlooked the campus with reflecting foil. But as the TMRC people used the word, there was serious respect implied. While someone might call a clever connection between relays a "mere hack," it would be understood that, to qualify as a hack, the feat must be imbued with innovation, style, and technical virtuosity. Even though one might self-deprecatingly say he was "hacking away at The System" (much as an axe-wielder hacks at logs), the artistry with which one hacked was recognized to

be considerable.

The most productive people working on Signals and Power called themselves "hackers" with great pride. Within the confines of the clubroom in Building 20, and of the "Tool Room" (where some study and many techno bull sessions took place), they had unilaterally endowed themselves with the heroic attributes of Icelandic legend. This is how Peter Samson saw himself and his friends in a Sandburg-esque poem in the club newsletter:

Switch Thrower for the World, Fuze Tester, Maker of Routes, Player with the Railroads and the System's Advance Chopper; Grungy, hairy, sprawling, Machine of the Point-Function Line-o-lite: They tell me you are wicked and I believe them; for I have seen your painted light bulbs under the lucite luring the system coolies . . . Under the tower, dust all over the place, hacking with bifur- cated springs . . . Hacking even as an ignorant freshman acts who has never lost occupancy and has dropped out Hacking the M-Boards, for under its locks are the switches, and under its control the advance around the layout, Hacking! Hacking the grungy, hairy, sprawling hacks of youth; uncabled, frying diodes, proud to be Switch-thrower, Fuze- tester, Maker of Routes, Player with Railroads, and Advance Chopper to the System.

Whenever they could, Samson and the others would slip off to the EAM room with their plug boards, trying to use the machine to keep track of the switches underneath the layout. Just as important, they were seeing what the electromechanical counter could do, taking it to its limit.

That spring of 1959, a new course was offered at MIT. It was the first course in programming a computer that freshmen could take. The teacher was a distant man with a wild shock of hair and an equally unruly beard—John McCarthy. A master mathematician, McCarthy was a classically absent-minded professor; stories abounded about his habit of suddenly answering a question hours, sometimes even days after it was first posed to him. He would approach you in the hallway, and with no salutation would begin speaking in his robotically precise diction, as if the pause in conversation had been only a fraction of a second, and not a week. Most likely, his belated response would be brilliant.

McCarthy was one of a very few people working in an entirely new form of scientific inquiry with computers. The volatile and controversial nature of his field of study was obvious from the very arrogance of the name that McCarthy had bestowed upon it: Artificial Intelligence. This man actually thought that computers could be SMART. Even at such a science-intensive place as MIT, most people considered the thought ridiculous: they considered computers to be useful, if somewhat absurdly expensive, tools for number-crunching huge calculations and for devising missile defense systems (as MIT's largest computer, the Whirlwind, had done for the early-warning SAGE system), but scoffed at the thought that computers themselves could actually be a scientific field of study, Computer Science did not officially exist at MIT in the late fifties, and McCarthy and his fellow computer specialists worked in the Electrical Engineering Department, which offered the course, No. 641, that Kotok, Samson, and a few other TRMC members took that spring.

McCarthy had started a mammoth program on the IBM 704—the Hulking Giant—that would give it the extraordinary ability to play chess. To critics of the budding field of Artificial Intelligence, this was just one example of the boneheaded optimism of people like John McCarthy. But McCarthy had a certain vision of what computers could do, and playing chess was only the beginning.

All fascinating stuff, but not the vision that was driving Kotok and Samson and the others. They wanted to learn how to WORK the damn machines, and while this new programming language called LISP that McCarthy was talking about in 641 was interesting, it was not nearly as interesting as the act of programming, or that fantastic moment when you got your printout back from the Priesthood—word from the source itself!—and could then spend hours poring over the results of the program, what had gone wrong with it, how it could be improved. The TMRC hackers were devising ways to get into closer contact with the IBM 704, which soon was upgraded to a newer model called the 709. By hanging out at the computation center in the wee hours of the morning, and by getting to know the Priesthood, and by bowing and scraping the requisite number of times, people like Kotok were eventually allowed to push a few buttons on the machine, and watch the lights as it worked.

There were secrets to those IBM machines that had been painstakingly learned by some of the older people at MIT with access to the 704 and friends among the Priesthood. Amazingly, a few of these programmers, grad students working with McCarthy, had even written a program that utilized one of the rows of tiny lights: the lights would be lit in such an order that it looked like a little ball was being passed from right to left: if an operator hit a switch at just the right time, the motion of the lights could be reversed—Computer Ping-Pong! This obviously was the kind of thing that you'd show off to impress your peers, who would then take a look at the actual program you had written and see how it was done.

To top the program, someone else might try to do the same thing with fewer instructions—a worthy

endeavor, since there was so little room in the small "memory" of the computers of those days that not many instructions could fit into them, John McCarthy had once noticed how his graduate students who loitered around the 704 would work over their computer programs to get the most out of the fewest instructions, and get the program compressed so that fewer cards would need to be fed to the machine. Shaving off an instruction or two was almost an obsession with them. McCarthy compared these students to ski bums. They got the same kind of primal thrill from "maximizing code" as fanatic skiers got from swooshing frantically down a hill. So the practice of taking a computer program and trying to cut off instructions without affecting the outcome came to be called "program bumming," and you would often hear people mumbling things like "Maybe I can bum a few instructions out and get the octal correction card loader down to three cards instead of four."

McCarthy in 1959 was turning his interest from chess to a new way of talking to the computer, the whole new "language" called LISP. Alan Kotok and his friends were more than eager to take over the chess project. Working on the batch-processed IBM, they embarked on the gargantuan project of teaching the 704, and later the 709, and even after that its replacement the 7090, how to play the game of kings. Eventually Kotok's group became the largest users of computer time in the entire MIT computation center.

Still, working with the IBM machine was frustrating. There was nothing worse than the long wait between the time you handed in your cards and the time your results were handed back to you. If you had misplaced as much as one letter in one instruction, the program would crash, and you would have to start the whole process over again. It went hand in hand with the stifling proliferation of goddamn RULES that permeated the atmosphere of the computation center. Most of the rules were designed to keep crazy young computer fans like Samson and Kotok and Saunders physically distant from the machine itself. The most rigid rule of all was that no one should be able to actually touch or tamper with the machine itself. This, of course, was what those Signals and Power people were dying to do more than anything else in the world, and the restrictions drove them mad.

One priest—a low-level sub-priest, really—on the late-night shift was particularly nasty in enforcing this rule, so Samson devised a suitable revenge. While poking around at Eli's electronic junk shop one day, he chanced upon an electrical board precisely like the kind of board holding the clunky vacuum tubes which resided inside the IBM. One night, sometime before 4 A.M., this particular sub-priest stepped out for a minute; when he returned, Samson told him that the machine wasn't working, but they'd found the trouble—and held up the totally smashed module from the old 704 he'd gotten at Eli's.

The sub-priest could hardly get the words out. "W-where did you get that?"

Samson, who had wide green eyes that could easily look maniacal, slowly pointed to an open place on the machine rack where, of course, no board had ever been, but the space still looked sadly bare. The sub-priest gasped. He made faces that indicated his bowels were about to give out. He whimpered exhortations to the deity. Visions, no doubt, of a million-dollar deduction from his paycheck began flashing before him. Only after his supervisor, a high priest with some understanding of the mentality of these young wiseguys from the Model Railroad Club, came and explained the situation did he calm down.

He was not the last administrator to feel the wrath of a hacker thwarted in the quest for access.

* * *

One day a former TMRC member who was now on the MIT faculty paid a visit to the clubroom. His name was Jack Dennis. When he had been an undergraduate in the early 1950s, he had worked furiously underneath the layout. Dennis lately had been working a computer which MIT had just received from Lincoln Lab, a military development laboratory affiliated with the Institute. The computer was called the TX-0, and it was one of the first transistor-run computers in the world. Lincoln Lab had used it specifically to test a giant computer called the TX-2, which had a memory so complex that only with this specially built little brother could its ills be capably diagnosed. Now that its original job was over, the three-million-dollar TX-0 had been shipped over to the Institute on "long-term loan," and apparently no one at Lincoln Lab had marked a calendar with a return date. Dennis asked the S&P people at TMRC whether they would like to see it.

Hey you nuns! Would you like to meet the Pope?

The TX-0 was in Building 26, in the second-floor Radio Laboratory of Electronics (RLE), directly above the first-floor Computation Center which housed the hulking IBM 704. The RLE lab resembled the control room of an antique spaceship. The TX-0, or Tixo, as it was sometimes called, was for its time a midget machine, since it was one of the first computers to use finger-size transistors instead of hand-size vacuum tubes. Still, it took up much of the room, along with its fifteen tons of supporting air-

conditioning equipment. The TX-O's workings were mounted on several tall, thin chassis, like rugged metal bookshelves, with tangled wires and neat little rows of tiny, bottle-like containers in which the transistors were inserted. Another rack had a solid metal front speckled with grim-looking gauges. Facing the racks was an L-shaped console, the control panel of this H. G. Wells spaceship, with a blue countertop for your elbows and papers. On the short arm of the L stood a Flexowriter, which resembled a typewriter converted for tank warfare, its bottom anchored in a military gray housing. Above the top were the control panels, boxlike protrusions painted an institutional yellow. On the sides of the boxes which faced the user were a few gauges, several lines of quarter-inch blinking lights, a matrix of steel toggle switches the size of large grains of rice, and, best of all, an actual cathode ray tube display, round and smoke-gray.

The TMRC people were awed. THIS MACHINE DID NOT USE CARDS. The user would first punch in a program onto a long, thin paper tape with a Flexowriter (there were a few extra Flexowriters in an adjoining room), then sit at the console, feed in the program by running the tape through a reader, and be able to sit there while the program ran. If something went wrong with the program, you knew immediately, and you could diagnose the problem by using some of the switches, or checking out which of the lights were blinking or lit. The computer even had an audio output: while the program ran, a speaker underneath the console would make a sort of music, like a poorly tuned electric organ whose notes would vibrate with a fuzzy, ethereal din. The chords on this "organ" would change, depending on what data the machine was reading at any given microsecond; after you were familiar with the tones, you could actually HEAR what part of your program the computer was working on. You would have to discern this, though, over the clacking of the Flexowriter, which could make you think you were in the middle of a machine-gun battle. Even more amazing was that, because of these "interactive" capabilities, and also because users seemed to be allowed blocks of time to use the TX-0 all by themselves, you could even modify a program WHILE SITTING AT THE COMPUTER. A miracle!

There was no way in hell that Kotok, Saunders, Samson, and the others were going to be kept away from that machine. Fortunately, there didn't seem to be the kind of bureaucracy surrounding the TX-0 that there was around the IBM 704. No cadre of officious priests. The technician in charge was a canny white-haired Scotsman named John McKenzie. While he made sure that graduate students and those working on funded projects— Officially Sanctioned Users—maintained access to the machine, McKenzie tolerated the crew of TMRC madmen who began to hang out in the RLE lab, where the TX-0 stood.

Samson, Kotok, Saunders, and a freshman named Bob Wagner soon figured out that the best time of all to hang out in Building 26 was at night, when no person in his right mind would have signed up for an hour-long session on the piece of paper posted every Friday beside the air conditioner in the RLE lab. The TX-0 as a rule was kept running twenty-four hours a day—computers back then were too expensive for their time to be wasted by leaving them idle through the night, and besides, it was a hairy procedure to get the thing up and running once it was turned off. So the TMRC hackers, who soon were referring to themselves as TX-0 hackers, changed their life-style to accommodate the computer. They laid claim to what blocks of time they could, and would "vulture time" with nocturnal visits to the lab on the off chance that someone who was scheduled for a 3 A.M. session might not show up.

"Oh!" Samson would say delightedly, a minute or so after someone failed to show up at the time designated in the logbook. "Make sure it doesn't go to waste!"

It never seemed to, because the hackers were there almost all the time. If they weren't in the RLE lab waiting for an opening to occur, they were in the classroom next to the TMRC clubroom, the Tool Room, playing a "hangman"-style word game that Samson had devised called "Come Next Door," waiting for a call from someone who was near the TX-0, monitoring it to see if someone had not shown up for a session. The hackers recruited a network of informers to give advance notice of potential openings at the computer—if a research project was not ready with its program in time, or a professor was sick, the word would be passed to TMRC and the hackers would appear at the TX-0, breathless and ready to jam into the space behind the console.

Though Jack Dennis was theoretically in charge of the operation, Dennis was teaching courses at the time, and preferred to spend the rest of his time actually writing code for the machine. Dennis played the role of benevolent godfather to the hackers: he would give them a brief hands-on introduction to the machine, point them in certain directions, be amused at their wild programming ventures. He had little taste for administration, though, and was just as happy to let John McKenzie run things. McKenzie early on recognized that the interactive nature of the TX-0 was inspiring a new form of computer programming, and the hackers were its pioneers. So he did not lay down too many edicts.

The atmosphere was loose enough in 1959 to accommodate the strays—science-mad people whose curiosity burned like a hunger, who like Peter Samson would be exploring the uncharted maze of

laboratories at MIT. The noise of the air-conditioning, the audio output, and the drill-hammer Flexowriter would lure these wanderers, who'd poke their heads into the lab like kittens peering into baskets of yarn.

One of those wanderers was an outsider named Peter Deutsch. Even before discovering the TX-0, Deutsch had developed a fascination for computers. It began one day when he picked up a manual that someone had discarded, a manual for an obscure form of computer language for doing calculations. Something about the orderliness of the computer instructions appealed to him: he would later describe the feeling as the same kind of eerily transcendent recognition that an artist experiences when he discovers the medium that is absolutely right for him. THIS IS WHERE I BELONG. Deutsch tried writing a small program, and, signing up for time under the name of one of the priests, ran it on a computer. Within weeks, he had attained a striking proficiency in programming. He was only twelve years old.

He was a shy kid, strong in math and unsure of most everything else. He was uncomfortably overweight, deficient in sports, but an intellectual star performer. His father was a professor at MIT, and Peter used that as his entree to explore the labs.

It was inevitable that he would be drawn to the TX-0. He first wandered into the small "Kluge Room" (a "kluge" is a piece of inelegantly constructed equipment that seems to defy logic by working properly), where three off-line Flexowriters were available for punching programs onto paper tape which would later be fed into the TX-0. Someone was busy punching in a tape. Peter watched for a while, then began bombarding the poor soul with questions about that weird-looking little computer in the next room. Then Peter went up to the TX-0 itself, examined it closely, noting how it differed from other computers: it was smaller, had a CRT display, and other neat toys. He decided right then to act as if he had a perfect right to be there. He got hold of a manual and soon was startling people by spouting actual make-sense computer talk, and eventually was allowed to sign up for night and weekend sessions, and to write his own programs.

McKenzie worried that someone might accuse him of running some sort of summer camp, with this short-pants little kid, barely tall enough to stick his head over the TX-O's console, staring at the code that an Officially Sanctioned User, perhaps some self-important graduate student, would be hammering into the Flexowriter, and saying in his squeaky, preadolescent voice something like "Your problem is that this credit is wrong over here . . . you need this other instruction over there," and the self-important grad student would go crazy—WHO IS THIS LITTLE WORM?—and start screaming at him to go out and play somewhere. Invariably, though, Peter Deutsch's comments would turn out to be correct. Deutsch would also brazenly announce that he was going to write better programs than the ones currently available, and he would go and do it.

Samson, Kotok, and the other hackers accepted Peter Deutsch: by virtue of his computer knowledge he was worthy of equal treatment. Deutsch was not such a favorite with the Officially Sanctioned Users, especially when he sat behind them ready to spring into action when they made a mistake on the Flexowriter. These Officially Sanctioned Users appeared at the TX-0 with the regularity of commuters. The programs they ran were statistical analyses, cross correlations, simulations of an interior of the nucleus of a cell. Applications. That was fine for Users, but it was sort of a waste in the minds of the hackers. What hackers had in mind was getting behind the console of the TX-0 much in the same way as getting in behind the throttle of a plane, Or, as Peter Samson, a classical music fan, put it, computing with the TX-0 was like playing a musical instrument: an absurdly expensive musical instrument upon which you could improvise, compose, and, like the beatniks in Harvard Square a mile away, wail like a banshee with total creative abandon.

One thing that enabled them to do this was the programming system devised by Jack Dennis and another professor, Tom Stockman. When the TX-0 arrived at MIT, it had been stripped down since its days at Lincoln Lab: the memory had been reduced considerably, to 4,096 "words" of eighteen bits each. (A "bit" is a BInary digiT, either a one or zero. These binary numbers are the only thing computers understand. A series of binary numbers is called a "word.") And the TX-0 had almost no software. So Jack Dennis, even before he introduced the TMRC people to the TX-0, had been writing "systems programs"—the software to help users utilize the machine.

The first thing Dennis worked on was an assembler. This was something that translated assembly language—which used three- letter symbolic abbreviations that represented instructions to the machine—into machine language, which consisted of the binary numbers 0 and 1. The TX-0 had a rather limited assembly language: since its design allowed only two bits of each eighteen-bit word to be used for instructions to the computer, only four instructions could be used (each possible two-bit variation—00, 0 1, 10, and 11—represented an instruction). Everything the computer did could be broken down to the execution of one of those four instructions: it took one instruction to add two numbers, but a series of

perhaps twenty instructions to multiply two numbers. Staring at a long list of computer commands written as binary numbers—for example, 10011001100001— could make you into a babbling mental case in a matter of minutes. But the same command in assembly language might look like this: ADD Y. After loading the computer with the assembler that Dennis wrote, you could write programs in this simpler symbolic form, and wait smugly while the computer did the translation into binary for you, Then you'd feed that binary "object" code back into the computer. The value of this was incalculable: it enabled programmers to write in something that LOOKED like code, rather than an endless, dizzying series of ones and zeros.

The other program that Dennis worked on with Stockman was something even newer—a debugger. The TX-0 came with a debugging program called UT-3, which enabled you to talk to the computer while it was running by typing commands directly into the Flexowriter, But it had terrible problems-for one thing, it only accepted typed-in code that used the octal numeric system. "Octal" is a base-eight number system (as opposed to binary, which is base two, and Arabic—ours-which is base ten), and it is a difficult system to use. So Dennis and Stockman decided to write something better than UT-3 which would enable users to use the symbolic, easier-to-work-with assembly language. This came to be called FLIT, and it allowed users to actually find program bugs during a session, fix them, and keep the program running. (Dennis would explain that "FLIT" stood for FLexowriter Interrogation Tape, but clearly the name's real origin was the insect spray with that brand name.) FLIT was a quantum leap forward, since it liberated programmers to actually do original composing on the machine—just like musicians composing on their musical instruments. With the use of the debugger, which took up one third of the 4,096 words of the TX-O's memory, hackers were free to create a new, more daring style of programming.

And what did these hacker programs DO? Well, sometimes, it didn't matter much at all what they did. Peter Samson hacked the night away on a program that would instantly convert Arabic numbers to Roman numerals, and Jack Dennis, after admiring the skill with which Samson had accomplished this feat, said, "My God, why would anyone want to do such a thing?" But Dennis knew why. There was ample justification in the feeling of power and accomplishment Samson got when he fed in the paper tape, monitored the lights and switches, and saw what were once plain old blackboard Arabic numbers coming back as the numerals the Romans had hacked with.

In fact it was Jack Dennis who suggested to Samson that there were considerable uses for the TX-O's ability to send noise to the audio speaker. While there were no built-in controls for pitch, amplitude, or tone character, there was a way to control the speaker—sounds would be emitted depending on the state of the fourteenth bit in the eighteen-bit words the TX-0 had in its accumulator in a given microsecond. The sound was on or off depending on whether bit fourteen was a one or zero. So Samson set about writing programs that varied the binary numbers in that slot in different ways to produce different pitches.

At that time, only a few people in the country had been experimenting with using a computer to output any kind of music, and the methods they had been using required massive computations before the machine would so much as utter a note, Samson, who reacted with impatience to those who warned he was attempting the impossible, wanted a computer playing music right away. So he learned to control that one bit in the accumulator so adeptly that he could command it with the authority of Charlie Parker on the saxophone. In a later version of this music compiler, Samson rigged it so that if you made an error in your programming syntax, the Flexowriter would switch to a red ribbon and print "To err is human to forgive divine."

When outsiders heard the melodies of Johann Sebastian Bach in a single-voice, monophonic square wave, no harmony, they were universally unfazed. Big deal! Three million dollars for this giant hunk of machinery, and why shouldn't it do at least as much as a five-dollar toy piano? It was no use to explain to these outsiders that Peter Samson had virtually bypassed the process by which music had been made for eons. Music had always been made by directly creating vibrations that were sound. What happened in Samson's program was that a load of numbers, bits of information fed into a computer, comprised a code in which the music resided. You could spend hours staring at the code, and not be able to divine where the music was. It only became music while millions of blindingly brief exchanges of data were taking place in the accumulator sitting in one of the metal, wire, and silicon racks that comprised the TX-0. Samson had asked the computer, which had no apparent knowledge of how to use a voice, to lift itself in song—and the TX-0 had complied.

So it was that a computer program was not only metaphorically a musical composition—it was LITERALLY a musical composition! It looked like—and was—the same kind of program which yielded complex arithmetical computations and statistical analyses. These digits that Samson had jammed into the computer were a universal language which could produce ANYTHING—a Bach fugue or an anti-aircraft system.

Samson did not say any of this to the outsiders who were unimpressed by his feat. Nor did the hackers themselves discuss this—it is not even clear that they analyzed the phenomenon in such cosmic terms. Peter Samson did it, and his colleagues appreciated it, because it was obviously a neat hack. That was justification enough.

* * *

To hackers like Bob Saunders—balding, plump, and merry disciple of the TX-0, president of TMRC's S&P group, student of systems— it was a perfect existence. Saunders had grown up in the suburbs of Chicago, and for as long as he could remember the workings of electricity and telephone circuitry had fascinated him. Before beginning MIT, Saunders had landed a dream summer job, working for the phone company installing central office equipment, He would spend eight blissful hours with soldering iron and pliers in hand, working in the bowels of various systems, an idyll broken by lunch hours spent in deep study of phone company manuals. It was the phone company equipment underneath the TMRC layout that had convinced Saunders to become active in the Model Railroad Club.

Saunders, being an upperclassman, had come to the TX-0 later in his college career than Kotok and Samson: he had used the breathing space to actually lay the foundation for a social life, which included courtship of and eventual marriage to Marge French, who had done some non-hacking computer work for a research project. Still, the TX-0 was the center of his college career, and he shared the common hacker experience of seeing his grades suffer from missed classes. It didn't bother him much, because he knew that his real education was occurring in Room 240 of Building 26, behind the Tixo console. Years later he would describe himself and the others as "an elite group. Other people were off studying, spending their days up on four-floor buildings making obnoxious vapors or off in the physics lab throwing particles at things or whatever it is they do. And we were simply not paying attention to what other folks were doing because we had no interest in it. They were studying what they were studying and we were studying what we were studying. And the fact that much of it was not on the officially approved curriculum was by and large immaterial."

The hackers came out at night. It was the only way to take full advantage of the crucial "off-hours" of the TX-0. During the day, Saunders would usually manage to make an appearance in a class or two. Then some time spent performing "basic maintenance"—things like eating and going to the bathroom. He might see Marge for a while. But eventually he would filter over to Building 26. He would go over some of the programs of the night before, printed on the nine-and-a-half-inch-wide paper that the Flexowriter used. He would annotate and modify the listing to update the code to whatever he considered the next stage of operation. Maybe then he would move over to the Model Railroad Club, and he'd swap his program with someone, checking simultaneously for good ideas and potential bugs. Then back to Building 26, to the Kluge Room next to the TX-0, to find an off-line Flexowriter on which to update his code. All the while he'd be checking to see if someone had canceled a one-hour session on the machine; his own session was scheduled at something like two or three in the morning. He'd wait in the Kluge Room, or play some bridge back at the Railroad Club, until the time came.

Sitting at the console, facing the metal racks that held the computer's transistors, each transistor representing a location that either held or did not hold a bit of memory, Saunders would set up the Flexowriter, which would greet him with the word "WALRUS." This was something Samson had hacked, in honor of Lewis Carroll's poem with the line "The time has come, the Walrus said . . ." Saunders might chuckle at that as he went into the drawer for the paper tape which held the assembler program and fed that into the tape reader. Now the computer would be ready to assemble his program, so he'd take the Flexowriter tape he'd been working on and send that into the computer. He'd watch the lights go on as the computer switched his code from "source" (the symbolic assembly language) to "object" code (binary), which the computer would punch out into another paper tape. Since that tape was in the object code that the TX-0 understood, he'd feed it in, hoping that the program would run magnificently.

There would most probably be a few fellow hackers kibitzing behind him, laughing and joking and drinking Cokes and eating some junk food they'd extracted from the machine downstairs. Saunders preferred the lemon jelly wedges that the others called "lemon gunkies." But at four in the morning, anything tasted good. They would all watch as the program began to run, the lights going on, the whine from the speaker humming in high or low register depending on what was in Bit 14 in the accumulator, and the first thing he'd see on the CRT display after the program had been assembled and run was that the program had crashed. So he'd reach into the drawer for the tape with the FLIT debugger and feed THAT into the computer. The computer would then be a debugging machine, and he'd send the program back in. Now he could start trying to find out where things had gone wrong, and maybe if he was lucky he'd find out, and change things by putting in some commands by flicking some of the switches on the console in precise order, or hammering in some code on the Flexowriter. Once things got running—and it was always incredibly satisfying when something worked, when he'd made that roomful of transistors and wires and metal and electricity all meld together to create a precise output

that he'd devised—he'd try to add the next advance to it. When the hour was over—someone already itching to get on the machine after him—Saunders would be ready to spend the next few hours figuring out what the heck had made the program go belly-up.

The peak hour itself was tremendously intense, but during the hours before, and even during the hours afterward, a hacker attained a state of pure concentration. When you programmed a computer, you had to be aware of where all the thousands of bits of information were going from one instruction to the next, and be able to predict—and exploit—the effect of all that movement. When you had all that information glued to your cerebral being, it was almost as if your own mind had merged into the environment of the computer. Sometimes it took hours to build up to the point where your thoughts could contain that total picture, and when you did get to that point, it was such a shame to waste it that you tried to sustain it by marathon bursts, alternatively working on the computer or poring over the code that you wrote on one of the off-line Flexowriters in the Kluge Room. You would sustain that concentration by "wrapping around" to the next day.

Inevitably, that frame of mind spilled over to what random shards of existence the hackers had outside of computing. The knife-and-paintbrush contingent at TMRC were not pleased at all by the infiltration of Tixo-mania into the club: they saw it as a sort of Trojan horse for a switch in the club focus, from railroading to computing. And if you attended one of the club meetings held every Tuesday at five-fifteen, you could see the concern: the hackers would exploit every possible thread of parliamentary procedure to create a meeting as convoluted as the programs they were hacking on the TX-0. Motions were made to make motions to make motions, and objections ruled out of order as if they were so many computer errors. A note in the minutes of the meeting on November 24, 1959, suggests that "we frown on certain members who would do the club a lot more good by doing more S&P-ing and less reading Robert's Rules of Order." Samson was one of the worst offenders, and at one point, an exasperated TMRC member made a motion "to purchase a cork for Samson's oral diarrhea."

Hacking parliamentary procedure was one thing, but the logical mind-frame required for programming spilled over into more commonplace activities. You could ask a hacker a question and sense his mental accumulator processing bits until he came up with a precise answer to the question you asked. Marge Saunders would drive to the Safeway every Saturday morning in the Volkswagen and upon her return ask her husband, "Would you like to help me bring in the groceries?" Bob Saunders would reply, "No." Stunned, Marge would drag in the groceries herself. After the same thing occurred a few times, she exploded, hurling curses at him and demanding to know why he said no to her question.

"That's a stupid question to ask," he said. "Of course I won't LIKE to help you bring in the groceries. If you ask me if I'll help you bring them in, that's another matter."

It was as if Marge had submitted a program into the TX-0, and the program, as programs do when the syntax is improper, had crashed. It was not until she debugged her question that Bob Saunders would allow it to run successfully on his own mental computer.

CHAPTER 2 THE HACKER ETHIC

Something new was coalescing around the TX-0: a new way of life, with a philosophy, an ethic, and a dream.

There was no one moment when it started to dawn on the TX-0 hackers that by devoting their technical abilities to computing with a devotion rarely seen outside of monasteries they were the vanguard of a daring symbiosis between man and machine. With a fervor like that of young hot-rodders fixated on souping up engines, they came to take their almost unique surroundings for granted, Even as the elements of a culture were forming, as legends began to accrue, as their mastery of programming started to surpass any previous recorded levels of skill, the dozen or so hackers were reluctant to acknowledge that their tiny society, on intimate terms with the TX-0, had been slowly and implicitly piecing together a body of concepts, beliefs, and mores.

The precepts of this revolutionary Hacker Ethic were not so much debated and discussed as silently agreed upon. No manifestos were issued. No missionaries tried to gather converts. The computer did the converting, and those who seemed to follow the Hacker Ethic most faithfully were people like Samson, Saunders, and Kotok, whose lives before MIT seemed to be mere preludes to that moment when they fulfilled themselves behind the console of the TX-0. Later there would come hackers who took the implicit Ethic even more seriously than the TX-0 hackers did, hackers like the legendary Greenblatt or Gosper, though it would be some years yet before the tenets of hackerism would be explicitly delineated.

Still, even in the days of the TX-0, the planks of the platform were in place. The Hacker Ethic:

ACCESS TO COMPUTERS—AND ANYTHING WHICH MIGHT TEACH YOU SOMETHING ABOUT THE WAY THE WORLD WORKS—SHOULD BE UNLIMITED AND TOTAL. ALWAYS YIELD TO THE HANDS-ON IMPERATIVE!

Hackers believe that essential lessons can be learned about the systems—about the world—from taking things apart, seeing how they work, and using this knowledge to create new and even more interesting things. They resent any person, physical barrier, or law that tries to keep them from doing this.

This is especially true when a hacker wants to fix something that (from his point of view) is broken or needs improvement. Imperfect systems infuriate hackers, whose primal instinct is to debug them. This is one reason why hackers generally hate driving cars—the system of randomly programmed red lights and oddly laid out one-way streets causes delays which are so goddamned UNNECESSARY that the impulse is to rearrange signs, open up traffic-light control boxes . . .redesign the entire system.

In a perfect hacker world, anyone pissed off enough to open up a control box near a traffic light and take it apart to make it work better should be perfectly welcome to make the attempt. Rules which prevent you from taking matters like that into your own hands are too ridiculous to even consider abiding by. This attitude helped the Model Railroad Club start, on an extremely informal basis, something called the Midnight Requisitioning Committee. When TMRC needed a set of diodes, or some extra relays, to build some new feature into The System, a few S&P people would wait until dark and find their way into the places where those things were kept. None of the hackers, who were as a rule scrupulously honest in other matters, seemed to equate this with "stealing." A willful blindness.

ALL INFORMATION SHOULD BE FREE.

If you don't have access to the information you need to improve things, how can you fix them? A free exchange of information particularly when the information was in the form of a computer program, allowed for greater overall creativity. When you were working on a machine like the TX-0, which came with almost no software, everyone would furiously write systems programs to make programming easier—Tools to Make Tools, kept in the drawer by the console for easy access by anyone using the machine. This prevented the dread, time-wasting ritual of reinventing the wheel: instead of everybody writing his own version of the same program, the best version would be available to everyone, and everyone would be free to delve into the code and improve on THAT. A world studded with feature-full programs, bummed to the minimum, debugged to perfection.

The belief, sometimes taken unconditionally, that information should be free was a direct tribute to the way a splendid computer, or computer program, works—the binary bits moving in the most straightforward, logical path necessary to do their complex job, What was a computer but something which benefited from a free flow of information? If, say, the accumulator found itself unable to get information from the input/output (i/o) devices like the tape reader or the switches, the whole system would collapse. In the hacker viewpoint, any system could benefit from that easy flow of information.

MISTRUST AUTHORITY—PROMOTE DECENTRALIZATION.

The best way to promote this free exchange of information is to have an open system, something which presents no boundaries between a hacker and a piece of information or an item of equipment that he needs in his quest for knowledge, improvement, and time on-line. The last thing you need is a bureaucracy. Bureaucracies, whether corporate, government, or university, are flawed systems, dangerous in that they cannot accommodate the exploratory impulse of true hackers. Bureaucrats hide behind arbitrary rules (as opposed to the logical algorithms by which machines and computer programs operate): they invoke those rules to consolidate power, and perceive the constructive impulse of hackers as a threat.

The epitome of the bureaucratic world was to be found at a very large company called International Business Machines—IBM. The reason its computers were batch-processed Hulking Giants was only partially because of vacuum tube technology, The real reason was that IBM was a clumsy, hulking company which did not understand the hacking impulse. If IBM had its way (so the TMRC hackers thought), the world would be batch-processed, laid out on those annoying little punch cards, and only the most privileged of priests would be permitted to actually interact with the computer.

All you had to do was look at someone in the IBM world, and note the button-down white shirt, the neatly pinned black tie, the hair carefully held in place, and the tray of punch cards in hand. You could wander into the Computation Center, where the 704, the 709, and later the 7090 were stored—the best IBM had to offer—and see the stifling orderliness, down to the roped-off areas beyond which non-authorized people could not venture. And you could compare that to the extremely informal atmosphere around the TX-0, where grungy clothes were the norm and almost anyone could wander in.

Now, IBM had done and would continue to do many things to advance computing. By its sheer size and mighty influence, it had made computers a permanent part of life in America. To many people, the words IBM and computer were virtually synonymous. IBM's machines were reliable workhorses, worthy of the trust that businessmen and scientists invested in them. This was due in part to IBM's conservative approach: it would not make the most technologically advanced machines, but would rely on proven concepts and careful, aggressive marketing. As IBM's dominance of the computer field was established, the company became an empire unto itself, secretive and smug.

What really drove the hackers crazy was the attitude of the IBM priests and sub-priests, who seemed to think that IBM had the only "real" computers, and the rest were all trash. You couldn't talk to those people—they were beyond convincing. They were batch-processed people, and it showed not only in their preference of machines, but in their idea about the way a computation center, and a world, should be run. Those people could never understand the obvious superiority of a decentralized system, with no one giving orders: a system where people could follow their interests, and if along the way they discovered a flaw in the system, they could embark on ambitious surgery. No need to get a requisition form. just a need to get something done.

This antibureaucratic bent coincided neatly with the personalities of many of the hackers, who since childhood had grown accustomed to building science projects while the rest of their classmates were banging their heads together and learning social skills on the field of sport. These young adults who were once outcasts found the computer a fantastic equalizer, experiencing a feeling, according to Peter Samson, "like you opened the door and walked through this grand new universe . . ." Once they passed through that door and sat behind the console of a million-dollar computer, hackers had power. So it was natural to distrust any force which might try to limit the extent of that power.

HACKERS SHOULD BE JUDGED BY THEIR HACKING, NOT BOGUS CRITERIA SUCH AS DEGREES, AGE, RACE, OR POSITION.

The ready acceptance of twelve-year-old Peter Deutsch in the TX-0 community (though not by non-hacker graduate students) was a good example. Likewise, people who trotted in with seemingly impressive credentials were not taken seriously until they proved themselves at the console of a computer. This meritocratic trait was not necessarily rooted in the inherent goodness of hacker hearts —it was mainly that hackers cared less about someone's superficial characteristics than they did about his potential to advance the general state of hacking, to create new programs to admire, to talk about that new feature in the system.

YOU CAN CREATE ART AND BEAUTY ON A COMPUTER.

Samson's music program was an example. But to hackers, the art of the program did not reside in the pleasing sounds emanating from the on-line speaker. The code of the program held a beauty of its own. (Samson, though, was particularly obscure in refusing to add comments to his source code explaining what he was doing at a given time. One well-distributed program Samson wrote went on for hundreds of assembly language instructions, with only one comment beside an instruction which contained the number 1750. The comment was RIPJSB, and people racked their brains about its meaning until someone figured out that 1750 was the year Bach died, and that Samson had written an abbreviation for Rest In Peace Johann Sebastian Bach.)

A certain esthetic of programming style had emerged. Because of the limited memory space of the TX-0 (a handicap that extended to all computers of that era), hackers came to deeply appreciate innovative techniques which allowed programs to do complicated tasks with very few instructions. The shorter a program was, the more space you had left for other programs, and the faster a program ran. Sometimes when you didn't need speed or space much, and you weren't thinking about art and beauty, you'd hack together an ugly program, attacking the problem with "brute force" methods. "Well, we can do this by adding twenty numbers," Samson might say to himself, "and it's quicker to write instructions to do that than to think out a loop in the beginning and the end to do the same job in seven or eight instructions." But the latter program might be admired by fellow hackers, and some programs were bummed to the fewest lines so artfully that the author's peers would look at it and almost melt with awe.

Sometimes program bumming became competitive, a macho contest to prove oneself so much in command of the system that one could recognize elegant shortcuts to shave off an instruction or two, or, better yet, rethink the whole problem and devise a new algorithm which would save a whole block of instructions. (An algorithm is a specific procedure which one can apply to solve a complex computer problem; it is sort of a mathematical skeleton key.) This could most emphatically be done by approaching the problem from an offbeat angle that no one had ever thought of before but that in retrospect made total sense. There was definitely an artistic impulse residing in those who could utilize

this genius-from-Mars techniques black-magic, visionary quality which enabled them to discard the stale outlook of the best minds on earth and come up with a totally unexpected new algorithm.

This happened with the decimal print routine program. This was a subroutines program within a program that you could sometimes integrate into many different programs—to translate binary numbers that the computer gave you into regular decimal numbers. In Saunders' words, this problem became the "pawn's ass of programming—if you could write a decimal print routine which worked you knew enough about the computer to call yourself a programmer of sorts." And if you wrote a GREAT decimal print routine, you might be able to call yourself a hacker. More than a competition, the ultimate bumming of the decimal print routine became a sort of hacker Holy Grail.

Various versions of decimal print routines had been around for some months. If you were being deliberately stupid about it, or if you were a genuine moron—an out-and-out "loser"—it might take you a hundred instructions to get the computer to convert machine language to decimal. But any hacker worth his salt could do it in less, and finally, by taking the best of the programs, bumming an instruction here and there, the routine was diminished to about fifty instructions.

After that, things got serious. People would work for hours, seeking a way to do the same thing in fewer lines of code. It became more than a competition; it was a quest. For all the effort expended, no one seemed to be able to crack the fifty-line barrier. The question arose whether it was even possible to do it in less. Was there a point beyond which a program could not be bummed?

Among the people puzzling with this dilemma was a fellow named Jenson, a tall, silent hacker from Maine who would sit quietly in the Kluge Room and scribble on printouts with the calm demeanor of a backwoodsman whittling. Jenson was always looking for ways to compress his programs in time and space—his code was a completely bizarre sequence of intermingled Boolean and arithmetic functions, often causing several different computations to occur in different sections of the same eighteen-bit "word." Amazing things, magical stunts.

Before Jenson, there had been general agreement that the only logical algorithm for a decimal print routine would have the machine repeatedly subtracting, using a table of the powers of ten to keep the numbers in proper digital columns. Jenson somehow figured that a powers-of-ten table wasn't necessary; he came up with an algorithm that was able to convert the digits in a reverse order but, by some digital sleight of hand, print them out in the proper order. There was a complex mathematical justification to it that was clear to the other hackers only when they saw Jenson's program posted on a bulletin board, his way of telling them that he had taken the decimal print routine to its limit. FORTY-SIX INSTRUCTIONS. People would stare at the code and their jaws would drop. Marge Saunders remembers the hackers being unusually quiet for days afterward.

"We knew that was the end of it," Bob Saunders later said. "That was Nirvana."

COMPUTERS CAN CHANGE YOUR LIFE FOR THE BETTER.

This belief was subtly manifest. Rarely would a hacker try to impose a view of the myriad advantages of the computer way of knowledge to an outsider. Yet this premise dominated the everyday behavior of the TX-0 hackers, as well as the generations of hackers that came after them.

Surely the computer had changed THEIR lives, enriched their lives, given their lives focus, made their lives adventurous. It had made them masters of a certain slice of fate. Peter Samson later said, "We did it twenty-five to thirty percent for the sake of doing it because it was something we could do and do well, and sixty percent for the sake of having something which was in its metaphorical way alive, our offspring, which would do things on its own when we were finished. That's the great thing about programming, the magical appeal it has . . . Once you fix a behavioral problem [a computer or program] has, it's fixed forever, and it is exactly an image of what you meant."

LIKE ALADDIN'S LAMP, YOU COULD GET IT TO DO YOUR BIDDING.

Surely everyone could benefit from experiencing this power. Surely everyone could benefit from a world based on the Hacker Ethic. This was the implicit belief of the hackers, and the hackers irreverently extended the conventional point of view of what computers could and should do—leading the world to a new way of looking and interacting with computers.

This was not easily done. Even at such an advanced institution as MIT, some professors considered a manic affinity for computers as frivolous, even demented. TMRC hacker Bob Wagner once had to explain to an engineering professor what a computer was. Wagner experienced this clash of computer versus anti-computer even more vividly when he took a Numerical Analysis class in which the professor required each student to do homework using rattling, clunky electromechanical calculators. Kotok was

in the same class, and both of them were appalled at the prospect of working with those lo-tech machines. "Why should we," they asked, "when we've got this computer?"

So Wagner began working on a computer program that would emulate the behavior of a calculator. The idea was outrageous. To some, it was a misappropriation of valuable machine time. According to the standard thinking on computers, their time was too precious that one should only attempt things which took maximum advantage of the computer, things that otherwise would take roomfuls of mathematicians days of mindless calculating. Hackers felt otherwise: anything that seemed interesting or fun was fodder for computing—and using interactive computers, with no one looking over your shoulder and demanding clearance for your specific project, you could act on that belief. After two or three months of tangling with intricacies of floating-point arithmetic (necessary to allow the program to know where to place the decimal point) on a machine that had no simple method to perform elementary multiplication, Wagner had written three thousand lines of code that did the job. He had made a ridiculously expensive computer perform the function of a calculator that cost a thousand times less. To honor this irony, he called the program Expensive Desk Calculator, and proudly did the homework for his class on it.

His grade—zero. "You used a computer!" the professor told him.
"This CAN'T be right."

Wagner didn't even bother to explain. How could he convey to his teacher that the computer was making realities out of what were once incredible possibilities? Or that another hacker had even written a program called Expensive Typewriter that converted the TX-0 to something you could write text on, could process your writing in strings of characters and print it out on the Flexowriter—could you imagine a professor accepting a classwork report WRITTEN BY THE COMPUTER? How could that professor—how could, in fact, anyone who hadn't been immersed in this uncharted man-machine universe—understand how Wagner and his fellow hackers were routinely using the computer to simulate, according to Wagner, "strange situations which one could scarcely envision otherwise"? The professor would learn in time, as would everyone, that the world opened up by the computer was a limitless one.

If anyone needed further proof, you could cite the project that Kotok was working on in the Computation Center, the chess program that bearded Al professor "Uncle" John McCarthy, as he was becoming known to his hacker students, had begun on the IBM 704. Even though Kotok and the several other hackers helping him on the program had only contempt for the IBM batch-processing mentality that pervaded the machine and the people around it, they had managed to scrounge some late-night time to use it interactively, and had been engaging in an informal battle with the systems programmers on the 704 to see which group would be known as the biggest consumer of computer time. The lead would bounce back and forth, and the white-shirt-and-black-tie 704 people were impressed enough to actually let Kotok and his group touch the buttons and switches on the 704: rare sensual contact with a vaunted IBM beast.

Kotok's role in bringing the chess program to life was indicative of what was to become the hacker role in Artificial Intelligence: a Heavy Head like McCarthy or like his colleague Marvin Minsky would begin a project or wonder aloud whether something might be possible, and the hackers, if it interested them, would set about doing it.

The chess program had been started using FORTRAN, one of the early computer languages. Computer languages look more like English than assembly language, are easier to write with, and do more things with fewer instructions; however, each time an instruction is given in a computer language like FORTRAN, the computer must first translate that command into its own binary language. A program called a compiler does this, and the compiler takes up time to do its job, as well as occupying valuable space within the computer. In effect, using a computer language puts you an extra step away from direct contact with the computer, and hackers generally preferred assembly or, as they called it, "machine" language to less elegant, "higher-level" languages like FORTRAN.

Kotok, though, recognized that because of the huge amounts of numbers that would have to be crunched in a chess program, part of the program would have to be done in FORTRAN, and part in assembly. They hacked it part by part, with "move generators," basic data structures, and all kinds of innovative algorithms for strategy. After feeding the machine the rules for moving each piece, they gave it some parameters by which to evaluate its position, consider various moves, and make the move which would advance it to the most advantageous situation. Kotok kept at it for years, the program growing as MIT kept upgrading its IBM computers, and one memorable night a few hackers gathered to see the program make some of its first moves in a real game. Its opener was quite respectable, but after eight or so exchanges there was real trouble, with the computer about to be checkmated. Everybody wondered how the computer would react. It too a while (everyone knew that during those

pauses the computer was actually "thinking," if your idea of thinking included mechanically considering various moves, evaluating them, rejecting most, and using a predefined set of parameters to ultimately make a choice). Finally, the computer moved a pawn two squares forward—illegally jumping over another piece. A bug! But a clever one—it got the computer out of check. Maybe the program was figuring out some new algorithm with which to conquer chess.

At other universities, professors were making public proclamations that computers would never be able to beat a human being in chess. Hackers knew better. They would be the ones who would guide computers to greater heights than anyone expected. And the hackers, by fruitful, meaningful association with the computer, would be foremost among the beneficiaries.

But they would not be the only beneficiaries. Everyone could gain something by the use of thinking computers in an intellectually automated world. And wouldn't everyone benefit even more by approaching the world with the same inquisitive intensity, skepticism toward bureaucracy, openness to creativity, unselfishness in sharing accomplishments, urge to make improvements, and desire to build as those who followed the Hacker Ethic? By accepting others on the same unprejudiced basis by which computers accepted anyone who entered code into a Flexowriter? Wouldn't we benefit if we learned from computers the means of creating a perfect system? If EVERYONE could interact with computers with the same innocent, productive, creative impulse that hackers did, the Hacker Ethic might spread through society like a benevolent ripple, and computers would indeed change the world for the better.

In the monastic confines of the Massachusetts Institute of Technology, people had the freedom to live out this dream—the hacker dream. No one dared suggest that the dream might spread. Instead, people set about building, right there at MIT, a hacker Xanadu the likes of which might never be duplicated.

Hackers, Heroes of the Computer Revolution, by Steven Levy
(C)1984 by Steven Levy

*** END OF THE PROJECT GUTENBERG EBOOK HACKERS, HEROES OF THE COMPUTER REVOLUTION. CHAPTERS 1 AND 2 ***

Updated editions will replace the previous one—the old editions will be renamed.

Creating the works from print editions not protected by U.S. copyright law means that no one owns a United States copyright in these works, so the Foundation (and you!) can copy and distribute it in the United States without permission and without paying copyright royalties. Special rules, set forth in the General Terms of Use part of this license, apply to copying and distributing Project Gutenberg™ electronic works to protect the PROJECT GUTENBERG™ concept and trademark. Project Gutenberg is a registered trademark, and may not be used if you charge for an eBook, except by following the terms of the trademark license, including paying royalties for use of the Project Gutenberg trademark. If you do not charge anything for copies of this eBook, complying with the trademark license is very easy. You may use this eBook for nearly any purpose such as creation of derivative works, reports, performances and research. Project Gutenberg eBooks may be modified and printed and given away—you may do practically ANYTHING in the United States with eBooks not protected by U.S. copyright law. Redistribution is subject to the trademark license, especially commercial redistribution.

START: FULL LICENSE
THE FULL PROJECT GUTENBERG LICENSE
PLEASE READ THIS BEFORE YOU DISTRIBUTE OR USE THIS WORK

To protect the Project Gutenberg™ mission of promoting the free distribution of electronic works, by using or distributing this work (or any other work associated in any way with the phrase “Project Gutenberg”), you agree to comply with all the terms of the Full Project Gutenberg™ License available with this file or online at www.gutenberg.org/license.

Section 1. General Terms of Use and Redistributing Project Gutenberg™ electronic works

1.A. By reading or using any part of this Project Gutenberg™ electronic work, you indicate that you have read, understand, agree to and accept all the terms of this license and intellectual property (trademark/copyright) agreement. If you do not agree to abide by all the terms of this agreement, you must cease using and return or destroy all copies of Project Gutenberg™ electronic works in your possession. If you paid a fee for obtaining a copy of or access to a Project Gutenberg™ electronic work and you do not agree to be bound by the terms of this agreement, you may obtain a refund from the person or entity to whom you paid the fee as set forth in paragraph 1.E.8.

1.B. “Project Gutenberg” is a registered trademark. It may only be used on or associated in any way with an electronic work by people who agree to be bound by the terms of this agreement. There are a few things that you can do with most Project Gutenberg™ electronic works even without complying with the full terms of this agreement. See paragraph 1.C below. There are a lot of things you can do with Project Gutenberg™ electronic works if you follow the terms of this agreement and help preserve free future access to Project Gutenberg™ electronic works. See paragraph 1.E below.

1.C. The Project Gutenberg Literary Archive Foundation (“the Foundation” or PGLAF), owns a compilation copyright in the collection of Project Gutenberg™ electronic works. Nearly all the individual works in the collection are in the public domain in the United States. If an individual work is unprotected by copyright law in the United States and you are located in the United States, we do not claim a right to prevent you from copying, distributing, performing, displaying or creating derivative works based on the work as long as all references to Project Gutenberg are removed. Of course, we hope that you will support the Project Gutenberg™ mission of promoting free access to electronic works by freely sharing Project Gutenberg™ works in compliance with the terms of this agreement for keeping the Project Gutenberg™ name associated with the work. You can easily comply with the terms of this agreement by keeping this work in the same format with its attached full Project Gutenberg™ License when you share it without charge with others.

This particular work is one of the few individual works protected by copyright law in the United States and most of the remainder of the world, included in the Project Gutenberg collection with the permission of the copyright holder. Information on the copyright owner for this particular work and the terms of use imposed by the copyright holder on this work are set forth at the beginning of this work.

1.D. The copyright laws of the place where you are located also govern what you can do with this work. Copyright laws in most countries are in a constant state of change. If you are outside the United States, check the laws of your country in addition to the terms of this agreement before downloading, copying, displaying, performing, distributing or creating derivative works based on this work or any other Project Gutenberg™ work. The Foundation makes no representations concerning the copyright status of any work in any country other than the United States.

1.E. Unless you have removed all references to Project Gutenberg:

1.E.1. The following sentence, with active links to, or other immediate access to, the full Project Gutenberg™ License must appear prominently whenever any copy of a Project Gutenberg™ work (any work on which the phrase “Project Gutenberg” appears, or with which the phrase “Project Gutenberg” is associated) is accessed, displayed, performed, viewed, copied or distributed:

> This eBook is for the use of anyone anywhere in the United States and most other parts of the world at no cost and with almost no restrictions whatsoever. You may copy it, give it away or re-use it under the terms of the Project Gutenberg License included with this eBook or online at www.gutenberg.org. If you are not located in the United States, you will have to check the laws of the country where you are located before using this eBook.

1.E.2. If an individual Project Gutenberg™ electronic work is derived from texts not protected by U.S. copyright law (does not contain a notice indicating that it is posted with permission of the copyright holder), the work can be copied and distributed to anyone in the United States without paying any fees or charges. If you are redistributing or providing access to a work with the phrase “Project Gutenberg” associated with or appearing on the work, you must comply either with the requirements of paragraphs 1.E.1 through 1.E.7 or obtain permission for the use of the work and the Project Gutenberg™ trademark as set forth in paragraphs 1.E.8 or 1.E.9.

1.E.3. If an individual Project Gutenberg™ electronic work is posted with the permission of the copyright holder, your use and distribution must comply with both paragraphs 1.E.1 through 1.E.7 and any additional terms imposed by the copyright holder. Additional terms will be linked to the Project Gutenberg™ License for all works posted with the permission of the copyright holder found at the beginning of this work.

1.E.4. Do not unlink or detach or remove the full Project Gutenberg™ License terms from this work, or any files containing a part of this work or any other work associated with Project Gutenberg™.

1.E.5. Do not copy, display, perform, distribute or redistribute this electronic work, or any part of this electronic work, without prominently displaying the sentence set forth in paragraph 1.E.1 with active links or immediate access to the full terms of the Project Gutenberg™ License.

1.E.6. You may convert to and distribute this work in any binary, compressed, marked up, nonproprietary or proprietary form, including any word processing or hypertext form. However, if you provide access to or distribute copies of a Project Gutenberg™ work in a format other than “Plain Vanilla ASCII” or other format used in the official version posted on the official Project Gutenberg™ website (www.gutenberg.org), you must, at no additional cost, fee or expense to the user, provide a copy, a means of exporting a copy, or a means of obtaining a copy upon request, of the work in its original “Plain Vanilla ASCII” or other form. Any alternate format must include the full Project Gutenberg™ License as specified in paragraph 1.E.1.

1.E.7. Do not charge a fee for access to, viewing, displaying, performing, copying or distributing

any Project Gutenberg™ works unless you comply with paragraph 1.E.8 or 1.E.9.

1.E.8. You may charge a reasonable fee for copies of or providing access to or distributing Project Gutenberg™ electronic works provided that:

- You pay a royalty fee of 20% of the gross profits you derive from the use of Project Gutenberg™ works calculated using the method you already use to calculate your applicable taxes. The fee is owed to the owner of the Project Gutenberg™ trademark, but he has agreed to donate royalties under this paragraph to the Project Gutenberg Literary Archive Foundation. Royalty payments must be paid within 60 days following each date on which you prepare (or are legally required to prepare) your periodic tax returns. Royalty payments should be clearly marked as such and sent to the Project Gutenberg Literary Archive Foundation at the address specified in Section 4, “Information about donations to the Project Gutenberg Literary Archive Foundation.”

- You provide a full refund of any money paid by a user who notifies you in writing (or by e-mail) within 30 days of receipt that s/he does not agree to the terms of the full Project Gutenberg™ License. You must require such a user to return or destroy all copies of the works possessed in a physical medium and discontinue all use of and all access to other copies of Project Gutenberg™ works.

- You provide, in accordance with paragraph 1.F.3, a full refund of any money paid for a work or a replacement copy, if a defect in the electronic work is discovered and reported to you within 90 days of receipt of the work.

- You comply with all other terms of this agreement for free distribution of Project Gutenberg™ works.

1.E.9. If you wish to charge a fee or distribute a Project Gutenberg™ electronic work or group of works on different terms than are set forth in this agreement, you must obtain permission in writing from the Project Gutenberg Literary Archive Foundation, the manager of the Project Gutenberg™ trademark. Contact the Foundation as set forth in Section 3 below.

1.F.

1.F.1. Project Gutenberg volunteers and employees expend considerable effort to identify, do copyright research on, transcribe and proofread works not protected by U.S. copyright law in creating the Project Gutenberg™ collection. Despite these efforts, Project Gutenberg™ electronic works, and the medium on which they may be stored, may contain “Defects,” such as, but not limited to, incomplete, inaccurate or corrupt data, transcription errors, a copyright or other intellectual property infringement, a defective or damaged disk or other medium, a computer virus, or computer codes that damage or cannot be read by your equipment.

1.F.2. LIMITED WARRANTY, DISCLAIMER OF DAMAGES - Except for the “Right of Replacement or Refund” described in paragraph 1.F.3, the Project Gutenberg Literary Archive Foundation, the owner of the Project Gutenberg™ trademark, and any other party distributing a Project Gutenberg™ electronic work under this agreement, disclaim all liability to you for damages, costs and expenses, including legal fees. YOU AGREE THAT YOU HAVE NO REMEDIES FOR NEGLIGENCE, STRICT LIABILITY, BREACH OF WARRANTY OR BREACH OF CONTRACT EXCEPT THOSE PROVIDED IN PARAGRAPH 1.F.3. YOU AGREE THAT THE FOUNDATION, THE TRADEMARK OWNER, AND ANY DISTRIBUTOR UNDER THIS AGREEMENT WILL NOT BE LIABLE TO YOU FOR ACTUAL, DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE OR INCIDENTAL DAMAGES EVEN IF YOU GIVE NOTICE OF THE POSSIBILITY OF SUCH DAMAGE.

1.F.3. LIMITED RIGHT OF REPLACEMENT OR REFUND - If you discover a defect in this electronic work within 90 days of receiving it, you can receive a refund of the money (if any) you paid for it by sending a written explanation to the person you received the work from. If you received the work on a physical medium, you must return the medium with your written explanation. The person or entity that provided you with the defective work may elect to provide a replacement copy in lieu of a refund. If you received the work electronically, the person or entity providing it to you may choose to give you a second opportunity to receive the work electronically in lieu of a refund. If the second copy is also defective, you may demand a refund in writing without further opportunities to fix the problem.

1.F.4. Except for the limited right of replacement or refund set forth in paragraph 1.F.3, this work is provided to you ‘AS-IS’, WITH NO OTHER WARRANTIES OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PURPOSE.

1.F.5. Some states do not allow disclaimers of certain implied warranties or the exclusion or limitation of certain types of damages. If any disclaimer or limitation set forth in this agreement violates the law of the state applicable to this agreement, the agreement shall be interpreted to make the maximum disclaimer or limitation permitted by the applicable state law. The invalidity or unenforceability of any provision of this agreement shall not void the remaining provisions.

1.F.6. INDEMNITY - You agree to indemnify and hold the Foundation, the trademark owner, any agent or employee of the Foundation, anyone providing copies of Project Gutenberg™ electronic works in accordance with this agreement, and any volunteers associated with the production,

promotion and distribution of Project Gutenberg™ electronic works, harmless from all liability, costs and expenses, including legal fees, that arise directly or indirectly from any of the following which you do or cause to occur: (a) distribution of this or any Project Gutenberg™ work, (b) alteration, modification, or additions or deletions to any Project Gutenberg™ work, and (c) any Defect you cause.

Section 2. Information about the Mission of Project Gutenberg™

Project Gutenberg™ is synonymous with the free distribution of electronic works in formats readable by the widest variety of computers including obsolete, old, middle-aged and new computers. It exists because of the efforts of hundreds of volunteers and donations from people in all walks of life.

Volunteers and financial support to provide volunteers with the assistance they need are critical to reaching Project Gutenberg™'s goals and ensuring that the Project Gutenberg™ collection will remain freely available for generations to come. In 2001, the Project Gutenberg Literary Archive Foundation was created to provide a secure and permanent future for Project Gutenberg™ and future generations. To learn more about the Project Gutenberg Literary Archive Foundation and how your efforts and donations can help, see Sections 3 and 4 and the Foundation information page at www.gutenberg.org.

Section 3. Information about the Project Gutenberg Literary Archive Foundation

The Project Gutenberg Literary Archive Foundation is a non-profit 501(c)(3) educational corporation organized under the laws of the state of Mississippi and granted tax exempt status by the Internal Revenue Service. The Foundation's EIN or federal tax identification number is 64-6221541. Contributions to the Project Gutenberg Literary Archive Foundation are tax deductible to the full extent permitted by U.S. federal laws and your state's laws.

The Foundation's business office is located at 809 North 1500 West, Salt Lake City, UT 84116, (801) 596-1887. Email contact links and up to date contact information can be found at the Foundation's website and official page at www.gutenberg.org/contact

Section 4. Information about Donations to the Project Gutenberg Literary Archive Foundation

Project Gutenberg™ depends upon and cannot survive without widespread public support and donations to carry out its mission of increasing the number of public domain and licensed works that can be freely distributed in machine-readable form accessible by the widest array of equipment including outdated equipment. Many small donations ($1 to $5,000) are particularly important to maintaining tax exempt status with the IRS.

The Foundation is committed to complying with the laws regulating charities and charitable donations in all 50 states of the United States. Compliance requirements are not uniform and it takes a considerable effort, much paperwork and many fees to meet and keep up with these requirements. We do not solicit donations in locations where we have not received written confirmation of compliance. To SEND DONATIONS or determine the status of compliance for any particular state visit www.gutenberg.org/donate.

While we cannot and do not solicit contributions from states where we have not met the solicitation requirements, we know of no prohibition against accepting unsolicited donations from donors in such states who approach us with offers to donate.

International donations are gratefully accepted, but we cannot make any statements concerning tax treatment of donations received from outside the United States. U.S. laws alone swamp our small staff.

Please check the Project Gutenberg web pages for current donation methods and addresses. Donations are accepted in a number of other ways including checks, online payments and credit card donations. To donate, please visit: www.gutenberg.org/donate

Section 5. General Information About Project Gutenberg™ electronic works

Professor Michael S. Hart was the originator of the Project Gutenberg™ concept of a library of electronic works that could be freely shared with anyone. For forty years, he produced and distributed Project Gutenberg™ eBooks with only a loose network of volunteer support.

Project Gutenberg™ eBooks are often created from several printed editions, all of which are confirmed as not protected by copyright in the U.S. unless a copyright notice is included. Thus, we do not necessarily keep eBooks in compliance with any particular paper edition.

Most people start at our website which has the main PG search facility: www.gutenberg.org.

This website includes information about Project Gutenberg™, including how to make donations to the Project Gutenberg Literary Archive Foundation, how to help produce our new eBooks, and how

to subscribe to our email newsletter to hear about new eBooks.