1 Instructions

1. Please write your name on the top right-hand corner of every sheet in this examination now.

2. This examination must be completed this evening. I will remain after class until 9:30 if anyone wants to continue working. You must turn in your work when you leave whether you are finished or not.

3. This examination is to be completed individually.

4. You should write your answers directly on these examination sheets. If there is not enough room, turn the sheets over and write on the back. If you do, please indicate that you have done so.

5. Read each question carefully. Many of them are only slightly different than the Review Questions. You should be able to answer them easily, but don’t assume you can use the same answers unchanged.

6. This examination has several parts. To score 200 points (“100 percent”) on the exam, you must answer at least some questions in each part. The table below shows the minimum questions you must answer from each part. (Note that merely answering the minimum number of questions from each section won’t get you a good grade. You must answer more than the minimum in at least one section.)
<table>
<thead>
<tr>
<th>Part</th>
<th>Subject</th>
<th>Minimum Number of Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>General knowledge and short-answer questions</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>Number Systems</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>Algorithms and Pseudocode</td>
<td>1</td>
</tr>
</tbody>
</table>

7. This examination is worth a maximum of 200 points toward your grades, but it contains questions with a total value of more than 200 points. Except for the requirement to solve a minimum number of problems from each part, you may work on the questions you wish to work on and ignore the rest. To get the best results, I suggest you scan the examination quickly to find the questions to which you know the answer and complete them. You should then go back and work on the questions of which you are not sure. When I grade the examination, you will get either the number of points you earned or 200 points whichever is lesser. However, I will keep your raw score and I may consider any extra points you may earn if I decide to curve the final grades.

2 General Knowledge and Short-Answer Questions

2.1 Notes

1. You must answer at least ten questions from this part.

2. Some questions begin on one page and end on the next. Make sure you read all of each question.

2.2 Questions

1. (3 points) The number of your section is 701. TA’s name is Aimn Blbol. His email address is ablbol1@umbc.edu.

2. (6 points) The purpose of computing is insight, not numbers.

3. (6 points) Computer Science is the study of algorithms.
4. (3 points) Which of these passwords are good ones and which are bad?

<table>
<thead>
<tr>
<th>Good</th>
<th>Bad</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>“MyPassword”</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>Your last name spelled backwards.</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>“ComputerSci”</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>Your license plate number.</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>“h&amp;s0me” (“handsome”)</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>Your first and last initials followed by your birthday in numbers, e.g. “FD790812”</td>
</tr>
</tbody>
</table>

All of these passwords — except for h&s0me — are either likely to be guessed or are too easy to figure out.

5. (5 points) Major components of a computer include:

<table>
<thead>
<tr>
<th>Yes</th>
<th>No</th>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td></td>
<td>Buss</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>CPU</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>Cabinet</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Dilithium crystals</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Domain Name System</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>I/O devices.</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>IP address</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Keyboard</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Main memory (RAM)</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Master inverter</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Network Interface</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Power supply</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Router</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Secondary storage</td>
</tr>
</tbody>
</table>

6. (2 points) The most important difference in performance between main memory and secondary storage is:

(a) Power consumption

(b) Lower probability of memory leakage

(c) Most secondary storage devices are too big to fit into the CPU.

(d) **Speed of access; secondary storage is much slower.**

(e) Secondary storage is mostly bistate storage, but main memory is mostly tri-state.

(f) Main memory is almost always content-addressable while secondary storage must use differential addressing.
7. (2 points) If we wanted to compare people to computers, we might say that, in a computer, the buss was the _________ and the CPU was the _________.
   (a) Spinal cord, brain.
   (b) Nervous system, digestive tract.
   (c) Eyes, brain.
   (d) Brain, heart.
   (e) Brain, spinal cord.
   (f) Dilithium crystals, force.

8. (5 points) Describe, in one sentence, the role of the CPU in a computer. The CPU controls the computer.

9. (2 points) What part of the computer exercises overall control? The CPU.
   (See the question immediately above.)

10. (2 points) True or False: A computer CPU may be very complex, but it is fundamentally just a collection of fast ON/OFF switches.

11. (2 points) Within a single computer, the major components communicate using:
    (a) The main memory.
    (b) The CPU
    (c) The memory cross-bar.
    (d) The buss.
    (e) The network interface.
    (f) The inter-component bandwidth.

12. (2 points) Main memory is made up of 2-state devices.

13. (2 points) An ON/OFF switch is a 2-state device.

14. (2 points) In memory, one state stands for 0, the other for 1.
15. (2 points) Memory can hold any type of data that can be represented by a combination of 2 types of symbols — and only those types.

16. (5 points) A group of 8 bits is called a byte.

17. (2 points) Memory that loses its contents when power is turned off is called volatile memory.

18. (2 points) The acronym ROM stands for Read-Only Memory.

19. (2 points) ROM is almost always non-volatile.

20. (2 points) True or False: Any address in main memory can be accessed in the same amount of time. That’s why it’s called “random-access memory.” It’s “RAM” because it’s Random-access memory. Sequential-access memory would be too slow.

21. (1 point) True or False: To reduce access time and improve information assurance, RAM is usually accessed sequentially. It’s “RAM” because it’s Random-access memory. Sequential-access memory would be too slow.

22. (2 points) True or False: ROM is usually a form of RAM.

23. (2 points) Tape and disk are two forms of secondary storage. A disk is a random-access device while a tape drive is sequential (serial) access.

24. (6 points) Most secondary storage is non-volatile storage because it retains its data when power goes off.

25. (4 points) Name two common I/O devices: floppy disk and keyboard. speaker, mouse, tape, printer, modem

26. (5 points) What does the word “Protocol” mean? Define it briefly in one sentence. A ‘‘protocol’’ is a set of rules for communication.

27. (4 points) Match each term with its best description:

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 “full-duplex”</td>
<td>2 “one-way communications”</td>
</tr>
<tr>
<td>2 “half-duplex”</td>
<td>1 “two-way communications”</td>
</tr>
</tbody>
</table>
28. (4 points) The difference between “serial” and “parallel” communication is that in “serial” communications bits are sent one-by-one while in “parallel” communications they are sent in groups.

29. (9 points) How many bits are in:
   (a) a nybble 4
   (b) a byte 8
   (c) a word 16 (with 32 an acceptable alternative)

30. (3 points) “**booting**” a computer is another name for starting it.

31. (3 points) Usually, the last thing done in the computer’s start up sequence is to load the **operating system**.

32. (2 points) The **operating system** is a control program that manages all the resources of the computer on which it runs.

33. (2 points) When a computer seems to do several things at once, it is said to **multitask**.

34. (2 points) **True** or False: The operating system is responsible for scheduling what resources are available and which processes get to use them.

35. (3 points) When an operating system is managing several tasks, it cycles through all of the tasks giving each one a small portion of time. That small portion is sometimes called a **time-slice, time quantum**.

36. (2 points) When an operating system saves a running job to disk and loads another job, it is said to __________ one job for the other
   (a) swap
   (b) exchange
   (c) switch
   (d) flip
37. (2 points) If a computer is running an operating system which appears to support several users at the same time, the computer itself is really doing ________ job(s) at a time.

(a) one
(b) two
(c) three
(d) four
(e) more than four.

38. (2 points) The central part ("hard nugget") of an operating system is called the

(a) critical section
(b) kernel
(c) nugget
(d) CPU driver
(e) execution queue

39. (2 points) The part of the operating system that maintains information about files available on the system is called the:

(a) device drivers
(b) master database
(c) file allocation system
(d) file manager

40. (2 points) Software to communicate with a peripheral device or controller in order to translate general requests into specific steps for that device is called a ________.

41. (2 points) The part of the operating system that controls the use of main memory is called the ________.
42. (5 points) A system which gives the illusion of having extra memory by storing some of its physical (RAM) memory by writing it out to disk is said to have “virtual memory.”

43. (2 points) The part of the operating system that maintains a record of processes that are present, adds new processes, removes completed processes is called the **scheduler**.

44. (2 points) The part of the operating system that executes processes when they are scheduled and ready is called **dispatcher**.

45. (2 points) True or False: Each program is associated with exactly one process.

46. (2 points) The dynamic activity of executing a program is called a:

   (a) thread
   (b) module
   (c) execution unit
   (d) dynamically-linked header
   (e) **process**

47. (7 points) Write the number of each item in column one in the empty box in front of the corresponding item in column two:

<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Shell</td>
<td>7 manage processes</td>
</tr>
<tr>
<td>2 File Manager</td>
<td>5 manages main memory</td>
</tr>
<tr>
<td>3 Instruction Register</td>
<td>2 manages mass memory</td>
</tr>
<tr>
<td>4 Device Drivers</td>
<td>3 decode instructions</td>
</tr>
<tr>
<td>5 Memory manager</td>
<td>6 Keep track of current point of execution</td>
</tr>
<tr>
<td>6 Program Counter</td>
<td>1 interface to user</td>
</tr>
<tr>
<td>7 Scheduler and Dispatcher</td>
<td>4 communicate with peripherals</td>
</tr>
</tbody>
</table>

48. (5 points) “Everything in Unix is a **file**.”
49. (8 points) A **hierarchical** file system is one in which some files (called **directories**) can contain other ones.

50. (10 points) Indicate which of the following strings are valid Unix file names and which are not:

<table>
<thead>
<tr>
<th>Valid?</th>
<th>Invalid?</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>1999Balance <em>(Can’t start with a number)</em></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>AbsenteeList</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Annual Calendar <em>(Can’t have embedded spaces.)</em></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Income1998</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>L1m1tedSc0pe</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Permanent:records <em>(Can’t contain “;”)</em></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Radio<em>Antenna <em>(Can’t contain “</em>”)</em></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Summary/October <em>(Can’t contain “/”)</em></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>a.out</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>gcc</td>
</tr>
</tbody>
</table>

51. (18 points) Match the command in column 1 with the description in column 2.
<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 cat filename</td>
<td>2 change to home dir</td>
</tr>
<tr>
<td>2 cd</td>
<td>12 move (rename) a file</td>
</tr>
<tr>
<td>3 cd filename</td>
<td>4 copy a file</td>
</tr>
<tr>
<td>4 cp filename1 filename2</td>
<td>14 print working directory</td>
</tr>
<tr>
<td>5 grep string filename</td>
<td>9 get the manual for filename</td>
</tr>
<tr>
<td>6 head filename</td>
<td>18 count lines, words, chars</td>
</tr>
<tr>
<td>7 logout</td>
<td>16 show last lines</td>
</tr>
<tr>
<td>8 lpr filename</td>
<td>8 print something</td>
</tr>
<tr>
<td>9 man filename</td>
<td>6 show first lines</td>
</tr>
<tr>
<td>10 mkdir filename</td>
<td>13 change your password</td>
</tr>
<tr>
<td>11 more filename</td>
<td>1 concatenate a file to stdout</td>
</tr>
<tr>
<td>12 mv filename1 filename2</td>
<td>3 change directory</td>
</tr>
<tr>
<td>13 passwd</td>
<td>17 write file and pipe</td>
</tr>
<tr>
<td>14 pwd</td>
<td>7 terminate a login session</td>
</tr>
<tr>
<td>15 rm filename</td>
<td>11 show by screens</td>
</tr>
<tr>
<td>16 tail filename</td>
<td>5 find something</td>
</tr>
<tr>
<td>17 tee filename</td>
<td>10 make a new directory</td>
</tr>
<tr>
<td>18 wc filename</td>
<td>15 remove (delete) a file</td>
</tr>
</tbody>
</table>

52. (4 points) Assume that you have two programs, *prog1* and *prog2*, and one file, *file*. Write the commands you would use to:

(a) have *prog1* take input from *file*  
    \[prog1 < file\]

(b) put output from *prog1* in *file*, erasing any existing file with that name  
    \[prog1 > file\]

(c) add output from *prog1* to *file* without erasing what is already in the file  
    \[prog1 >> file\]

(d) use output from *prog1* as input to *prog2*  
    \[prog1 | prog2\]
53. (2 points) In Unix, there are two I/O streams that are open for all programs. Name them. **stdin (standard input), stdout (standard output)**

54. (2 points) Name one editor for Unix. **emacs, pico, vi, ed, xedit, ted, jot**

55. (2 points) Name one mail program for Unix. **mail, xmail, elm, pine, mh, vm**

56. (2 points) The number system having just two symbols, 0 and 1, is called the **binary** number system.

57. (2 points) The **arithmetic logic unit (alu)** is the computer’s mechanism for performing operations on data and coordinating the sequence of these operations.

58. (2 points) The CPU consists of two main parts: the **arithmetic logic unit** and the **control unit**.

59. (5 points) Which of the following statements about CPU registers are true and which are false.

<table>
<thead>
<tr>
<th>True?</th>
<th>False?</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td></td>
<td>Registers are temporary memory inside the CPU.</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Registers are used to hold partial results and CPU control data.</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>Registers are accessed via special lines on the memory buss. <em>(Not if they’re inside the CPU.)</em></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Registers are often specialized, with special registers designed for different functions.</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>Registers are very fast.</td>
</tr>
</tbody>
</table>

60. (7 points) Indicate which of the following functions are performed by a CPU control unit (CTL) and which by the arithmetic logic unit (ALU).
<table>
<thead>
<tr>
<th>CTL</th>
<th>ALU</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CU</td>
<td></td>
<td>Keeps track of location of current instruction</td>
</tr>
<tr>
<td>CU</td>
<td></td>
<td>Fetchs instructions from memory</td>
</tr>
<tr>
<td>CU</td>
<td></td>
<td>Decodes instruction</td>
</tr>
<tr>
<td>CU</td>
<td></td>
<td>Loads registers with data</td>
</tr>
<tr>
<td>CU</td>
<td></td>
<td>Keeps track of free registers</td>
</tr>
<tr>
<td>CU</td>
<td></td>
<td>Instructs ALU which operations are required.</td>
</tr>
<tr>
<td>ALU</td>
<td></td>
<td>Executes arithmetic and logic instructions from Control Unit.</td>
</tr>
</tbody>
</table>

61. (2 points) Two memory cells have the following contents, \([M1] = 7\), \([M2] = 2\). If we copy from \(M1\) to \(M2\), show the contents of each cell after the operation.

<table>
<thead>
<tr>
<th>Cell</th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>M2</td>
<td>2</td>
<td>7</td>
</tr>
</tbody>
</table>

62. (7 points) Match the term in column 1 with the correct definition in column 2.

<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Decode</td>
<td>7 Contains the memory address of the next instruction to be executed.</td>
</tr>
<tr>
<td>2 Execute</td>
<td>5 specifies an operation, like LOAD, ADD, JUMP or STORE</td>
</tr>
<tr>
<td>3 Fetch</td>
<td>4 Contains the instruction currently being executed</td>
</tr>
<tr>
<td>4 Instruction Register</td>
<td>6 provides more information about the instruction.</td>
</tr>
<tr>
<td>5 Opcode</td>
<td>3 get the next instruction from memory</td>
</tr>
<tr>
<td>6 Operand</td>
<td>1 determine the bit pattern in the instruction register</td>
</tr>
<tr>
<td>7 Program counter</td>
<td>2 perform the action requested by the instruction in the instruction register.</td>
</tr>
</tbody>
</table>

63. (2 points) During program execution the control unit continuously repeats this 3-step cycle
(a) dilithium crystals
(b) factor, cofactor, resolve
(c) fetch, decode, execute
(d) load, analyze, export
(e) prime, align, run
(f) read, run, write

64. (6 points) Fill-in the truth tables for

<table>
<thead>
<tr>
<th>or</th>
<th>T</th>
<th>F</th>
<th>and</th>
<th>T</th>
<th>F</th>
<th>xor</th>
<th>T</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

65. (4 points) “problem-solving is the process of transforming the description of a problem into the solution of that problem by using our knowledge of the problem domain and by relying on our ability to select and use appropriate problem-solving strategies, techniques, and tools.”

66. (9 points) An algorithm is a procedure with

(a) A finite sequence of unambiguous actions.
(b) A definite criterion for stopping.

67. (3 points) In an algorithm, all the steps must be unambiguous, well-defined.

68. (3 points) An algorithm must have a clear stopping, termination rule.

69. (6 points) Indicate with True or False which of the following reasons might cause you to write down an algorithm.

<table>
<thead>
<tr>
<th>T</th>
<th>F</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td></td>
<td>To recognize and understand the essence of a problem or a solution</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>For your own use in the future.</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Document the process so that others can solve the problem, even if they know very little about the principles behind how the solution was derived.</td>
</tr>
</tbody>
</table>
70. (4 points) An algorithm is a **unambiguous** set of executable instructions that directs a **terminating** activity.

71. (4 points) When something is represented by something else, such as when an algorithm is represented in pseudocode,

- **syntax** refers to the representation itself
- **semantics** refers to the concept represented

72. (3 points) An algorithm is any **finite (well-defined, unambiguous) sequence** of steps connecting a **problem** with its **solution**.

73. (3 points) The goal of design is to develop a series of steps with a logical order which, when applied to the **input, problem**, would produce the specified **output, solution**.

74. (2 points) True or **False**: Because it allows the maximum flexibility throughout all stages of development, “Design at the Keyboard” is an excellent approach to software design.

75. (2 points) **implementation** is the phase where the design is transformed into code.

76. (6 points) Name three steps of the software development life-cycle.

- analysis, design, implementation, testing, maintenance

77. (6 points) Indicate by checking True or False which of these phrases describe pseudocode:

<table>
<thead>
<tr>
<th>T</th>
<th>F</th>
<th>Phrase</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td></td>
<td>Subtle grammar</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>English like</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Semi-formal</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Indirect verbs</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Limited vocabulary</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Intended specifically to design and describe algorithms.</td>
</tr>
</tbody>
</table>
78. (5 points) The main purpose of a pseudo-code is to describe *algorithms* in a simple, easy-to-understand manner for readers, who may or may not be proficient in computer programming.

79. (9 points) Match each term with its description by putting the term’s number in the empty box in front of the corresponding description.

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Repetition</td>
</tr>
<tr>
<td>3</td>
<td>Series of steps or statements that are executed in the order they are written.</td>
</tr>
<tr>
<td>2</td>
<td>Selection</td>
</tr>
<tr>
<td>1</td>
<td>Specifies a block of one or more statements that are repeatedly executed until a condition is satisfied.</td>
</tr>
<tr>
<td>3</td>
<td>Sequence</td>
</tr>
<tr>
<td>2</td>
<td>Defines one or two courses of action depending on the evaluation of a condition.</td>
</tr>
</tbody>
</table>

80. (6 points) All words in a pseudo-code statement must be chosen to be *clear, unambiguous*, and as *understandable* as possible to understand by non-programmers.

81. (6 points) When using pseudocode, enclose comments between `/*` and `*/`.

82. (10 points) In one or two sentences, write down the basic idea of top-down design (sometimes called “top-down step-wise refinement”).

The basic idea in "top-down step-wise refinement" is to take a large problem and break it up into smaller ones. The same procedure is then applied to the initial set of smaller problems to break them into yet smaller ones. This procedure is repeated until it yields a set of problems small enough to be solved easily. The procedure is then reversed to combine the solutions of smaller problems to yield the solution of the larger one from which they were obtained. The procedure is complete when the solution to the first original set of "smaller" problem are combined to yield a solution to the original problem.

83. (2 points) True or False: When developing a top-down design, the designer should place the highest value on keeping all the modules the same size.
84. (2 points) In the binary system negative numbers are usually written in two's-complement notation.

85. (2 points) The Decimal number system has 10 digits.

86. (2 points) The radix (base) of the hexadecimal system is 16.

87. (3 points) Write all the digits of the hexadecimal system.
   0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

3 Number Systems

3.1 Notes

1. You must answer at least four questions from this part.

2. Some questions begin on one page and end on the next. Make sure you read all of each question.

3.2 Questions

1. (5 points) Expand the following decimal numbers as sums of powers of powers of 10. Write the resulting coefficients in the appropriate columns below.

   | 10^4 | 10^3 | 10^2 | 10^1 | 10^0 |
---|------|------|------|------|------|
31416 | 3 | 1 | 4 | 1 | 6 |
55939 | 5 | 5 | 9 | 3 | 9 |
1024 | 1 | 0 | 2 | 4 |   |
23 |   |   | 2 | 3 |   |
16384 | 1 | 6 | 3 | 8 | 4 |

2. (5) These numbers are written explicitly as sums of powers of 10. Rewrite them in standard decimal format:

   \[ 3 \times 10^4 + 7 \times 10^3 + 0 \times 10^2 + 3 \times 10^1 + 7 \times 10^0 = 37037 \]
\[
\begin{align*}
4 \cdot 10^3 + 8 \cdot 10^2 + 1 \cdot 10^1 + 9 \cdot 10^0 &= 4819 \\
0 \cdot 10^4 + 6 \cdot 10^3 + 3 \cdot 10^2 + 0 \cdot 10^1 + 1 \cdot 10^0 &= 6301 \\
5 \cdot 10^4 + 3 \cdot 10^3 + 1 \cdot 10^2 + 6 \cdot 10^1 + 2 \cdot 10^0 &= 53162 \\
1 \cdot 10^4 + 2 \cdot 10^3 + 3 \cdot 10^2 + 4 \cdot 10^1 + 5 \cdot 10^0 &= 12345
\end{align*}
\]

3. (10 points) Translate the following binary numbers into decimal. You may write out your partial results.

(a) 111
(b) 110113
(c) 1101127
(d) 11011155
(e) 1101026

4. (10 points) Translate the following decimal numbers into binary

(a) 13 \hspace{1cm} 1101
(b) 14 \hspace{1cm} 1110
(c) 95 \hspace{1cm} 1011111 (This is one less than 96 = 64 + 32)
(d) 62 \hspace{1cm} 111110 (This is one less than 63)
(e) 23 \hspace{1cm} 10111

5. (20 points) Add the following sets of binary numbers

(a) 1011 + 110111000
(b) 110 + 1011011
(c) 110101 + 1001101110111
(d) $111 + 100 = 1011$

(e) $1001 + 1101 = 10110$

6. (5 points) Add the following numbers: $1010 \ 0011 \ 1011 \ 0001 + 0111 \ 0100 \ 0001 \ 1001 = \underline{1 \ 0001 \ 0111 \ 1100 \ 1010}$

7. (10 points) Translate the following decimal numbers into hexadecimal

(a) 21 \ 15

(b) 51 \ 33

(c) 39 \ 27

(d) 27 \ 1B

(e) 43 \ 2B

8. (10 points) Translate the following hexadecimal numbers into decimal

(a) 61 \ $6 \times 16 + 1 = 97$

(b) 17 \ $1 \times 16 + 7 = 23$

(c) 0F \ $0 \times 16 + 15 = 15$

(d) 62 \ $6 \times 16 + 2 = 98$

(e) F5 \ $15 \times 16 + 5 = 240 + 5 = 245$

9. (10 points) Translate the following hexadecimal numbers into binary

(a) 34 \ 0011 \ 0100 = 110100

(b) 45 \ 0100 \ 0101 = 1000101

(c) 57 \ 0101 \ 0111 = 1010111

(d) 23 \ 0010 \ 0011 = 100011

(e) 1B \ 0001 \ 1011 = 11011
10. (10 points) Convert the following binary numbers to their negatives

(a) $\begin{array}{c}
1 \\
0 + 1 = 1
\end{array}$

(b) $\begin{array}{c}
1101 \\
0010 + 1 = 0011
\end{array}$

(c) $\begin{array}{c}
11011 \\
00100 + 1 = 00101
\end{array}$

(d) $\begin{array}{c}
110111 \\
001000 + 1 = 001001
\end{array}$

(e) $\begin{array}{c}
11010 \\
00101 + 1 = 00110
\end{array}$

4 Algorithms and Pseudocode

4.1 Notes

1. You must answer at least one question from this part.

2. Some questions begin on one page and end on the next. Make sure you read all of each question.

3. In these problems I am looking for your ability to describe the overall design of the problem, not catalog all its details. While I would prefer pseudocode, I will accept clearly written algorithms in English or mathematical notation. Remember, the goal is to tell me how to do something and not merely to tell me “about” something.

   • If you tell me $a = 7$ you may be telling me about something, but
   • If you tell me “set $a$ equal to 7”, then you are clearly telling me to do something.

4.2 Questions

1. Imagine that I am a computer scientist who has just moved here from a foreign country and I don’t know how to make lemonade from frozen concentrate. Imagine that I’m standing next to the sink in my kitchen with an open can of (still frozen) concentrate, a pitcher, and a long-handled spoon in front of me. The lemonade uses four cans of water for each can of concentrate. My goal is to mix with the right amount of water, then stir the lemonade so that it is completely dissolved.
Write an algorithm in pseudocode telling me how to mix a pitcher of lemonade.

(a) open lemonade can
(b) dump concentrate into pitcher
(c) set cansOfWater to 0
(d) while cansOfWater is less than five
   (e) add canOfWater to pitcher
   (f) set cansOfWater to cansOfWater plus one
(g) end while
(h) grasp spoon
(i) set completelyMixed to false
(j) while completelyMixed is false
   (k) stir pitcher once
   (l) if pitcher contents are completely Mixed
   (m) set completelyMixed to true
   (n) end if
(o) end while
(p) complete

2. Imagine that I am a computer scientist who has just moved here from a foreign country and I don’t know how to use a hammer to drive a nail. I have a hammer, a long nail, and a thick block of wood on the bench before me. My goal is to hammer the nail until the head is flat against the top of the block of wood. (The block is thicker than the nail is long.)

(15 points) Write an algorithm in pseudocode telling me how to drive my nail.

(a) position wood
(b) grasp hammer in good hand (right if right-handed, left if left-handed)
(c) position nail on board with other hand
(d) start nail with one hammer stroke
(e) remove hand holding nail and grasp board clear of nail
(f) while nail head is not flat against wood surface
(g) strike nail squarely on head with hammer
(h) end_while
(i) complete

3. Imagine that I am a computer scientist who has just moved here from a foreign country and I don’t know how to use a coin-operated soda machine (such as the ones found in the passageway outside our classroom).

(a) (20 points) Write an algorithm in pseudocode telling me how me how to use a coin-operated soda machine.

(b) (15 points) Extend your algorithm to tell me how to use a machine that also takes dollar bills. (Make sure to explain what to do if the bill is rejected when I first put it in.)

Coin-operated machine

(a) read price from machine
(b) obtain coins
(c) if price_of_items is not more than than value_of_coins then
(d) select new item
(e) insert coins
(f) press selection button
(g) remove item
(h) if price_of_items is less than value_of_coins
(i) remove change
(j) end_if
(k) open_item
(l) else
(m) get more money or find less expensive machine
(n) end_if
Coin- and bill-operated machine

(a) read price from machine
(b) obtain coins and bills
(c) if price_of_items is more than value_of_coins_and_bills then
(d) quit, get more money or find less expensive machine
(e) end_if
(f) if price_of_items is not more than than value_of_coins then
(g) select new item
(h) while price_of_item is less than value_of_coins_inserted
(i) insert coin
(j) end_while
(k) else if price_of_items is not more than than value_of_bills then
(l) while price_of_item is less than value_of_bills_inserted
(m) insert bill
(n) end_while
(o) press selection button
(p) remove item
(q) if price_of_items is less than value_of_money_used
(r) remove change
(s) end_if
(t) open_item
(u) end_if

4. Finding the roots of a continuous function: If a continuous function, call it \( f(x) \) has a positive value for \( x = a \) and a negative value for \( x = b \) then there must be some point \( z \) such that \( a < z < b \) with \( f(z) = 0 \). The point \( z \) is called a zero of \( f \).

One way to look for \( z \) is to consider a new point \( c = (a + b)/2 \). If \( f(c) \) is close enough to zero, say \( |f(c)| < 0.01 \), then quit and use \( c \) as an approximation for \( z \). If not, then use \( c \) to replace whichever of \( a \) or \( b \) had the same sign as \( c \). One way to test this is to replace \( b \) if \( ac < 0 \) and to replace \( a \) if \( ab < 0 \). (Recall that if \( xy < 0 \) then \( x \) and
y must have opposite signs.) Of course, this same procedure can now be repeated with the new values of \(a\) and \(b\) and this repetition can be continued while \(|f(c)| > 0.01\) (or some other appropriate value).

(a) (35 points) Apply top-down design and write a pseudocode description of the following algorithm.

(b) (50 points) I’m a very impatient fellow. Let’s assume that I start with points \(a\) and \(b\) about 1024 units apart. How many times must I cycle through the loop of this algorithm before \(a\) and \(b\) are about 1 unit apart? If I want to know the answer within about \(\frac{1}{2}^{2048}\) of 1 unit, how many more times must I cycle through the algorithm?

**Pseudocode**

(a) input \(f\) /* might have to be defined in advance */
(b) set errtol 0.01
(c) input \(a\)
(d) while \(f(a)*f(b) > 0\)
(e) input \(b\) /* must have \(f(a)\) and \(f(b)\) opposite signs */
(f) end while
(g) while the absolute value of \(f(a)-f(b)\) is greater than errtol
(h) set \(c\) to \((a+b)/2\)
(i) if \(f(c)\) equals 0, then /* no real chance */
(j) set \(a\) to \(c\) /* insures stop this time thru loop */
(k) set \(b\) to \(c\)
(l) else if \(f(c)*f(b) < 0\)
(m) set \(a\) to \(c\)
(n) else if \(f(c)*f(a) < 0\)
(o) set \(b\) to \(c\)
(p) else
(q) error! /* this should never happen */
(r) end if
(s) end while
(t) return \(c\)
**Speed of Convergence**

The answer to this depends on understanding that the algorithm divides the interval of uncertainty *in half* with each cycle. The trial point, c, is always halfway between the two endpoints, a and b. So starting with an interval of 1024 and dividing successively by two, we get the series \{1024, 512, 256, 128, 64, 32, 16, 8, 4, 2, 1\} demonstrating that it will take 10 loops until a and b are about 1 unit apart. To answer the second part of the question, we can count the sequence \{1, 1/2, 1/4, \ldots 1/2048\} or we can observe that $2048 = 2 \ast 1024$ and conclude that it will take eleven more bisections to an interval of 1/2048.

5. There is an easy way to calculate the square root of a number with only a four-function pocket calculator. The only limitation is that the number must not be too big for the calculator to add, subtract, multiply, and divide.

The procedure is straight-forward. Call the number for which we’d like the square root \(N\). Take a guess at the square root and call that guess \(G\). Now divide \(N\) by \(G\) and call that number (the quotient) \(Q\). If \(G\) was a good guess and close to the square root of \(N\), then \(Q\) will also be close to the square root of \(N\), and so close to \(G\). If \(Q\) and \(G\) are “close enough,” then the square root of \(N\) will be even closer to their mean, \((Q + G)/2\). (Remember that if you divide a number by its square root, the result is also the square root, so if \(G\) is really the square root, then \(Q\) and \(G\) will be equal.) If they are not “close enough,” then update your guess \(G\) so it is equal to \((Q + G)/2\) and repeat the procedure. The process is guaranteed to converge eventually to the square root of \(N\).

(a) (35 points) Write a pseudo-code description of this algorithm.
(b) (90 points) The paragraph above says that this algorithm is “guaranteed” to converge. Why? Explain how we know convergence is guaranteed.

**Pseudocode**

(a) input f /* might have to be defined in advance */
(b) set errtol 0.01 /* this will work for "close enough" */
(c) set N to -1
(d) while $N$ less than zero /* can’t take real sqrt of negative number */
(e) input $N$
(f) end_while
(g) set $G$ to -1
(h) while $G$ less than zero /* want positive square root */
(i) input $G$
(j) end_while
(k) set $Q$ to $N/G$ /* $Q = G$ if $G$ is square root */
(l) while the absolute value of $Q-G$ is greater than errtol
(m) set $G$ to $(Q+G)/2$ /* try another guess */
(n) set $Q$ to $N/G$ /* get another quotient */
(o) end_while
(p) return $(Q+G)/2$ /* will be even closer than $Q$ or $G$ */

Why this procedure converges

The short answer is that convergence is guaranteed because the intervals between successive pairs of $G$ and $Q$ are "nested," meaning each one is completely enclosed by the one before. Imagine an infinite set of nested Russian dolls, each one smaller than the last. That’s the same idea. Because each interval is completely contained in the last, each must be smaller than the last. Eventually, the intervals will be as small as desired. (A formal proof can be constructed easily from the ‘‘Nested Intervals’’ theorem of advanced calculus, cf Angus Taylor, Advanced Calculus, Academic Press, NY, 1955, Theorem VI, page 95.)