CMPUT 101 - Solutions
Quiz 1 (50 minutes)
July 23, 2001

Last Name: ____________________________

First Name: ____________________________

Student ID: _____________

Section: B1

Instructor: I. E. Leonard

Instructions: Read carefully before proceeding.

No calculators, books or other aids are permitted for this test.

1. Write your name and ID number in the space provided above.
2. Wait until you are told to start working on the test (**don't turn the pages yet**).
3. Write your ID number on top of all remaining pages in the test, only then start answering the questions. Put all your answers on the test paper, no additional sheets of papers can be handed it. You can use the back of the pages for your scratch notes and calculations.
4. When you are told that the time is up, stop working on the test.

Good luck!

Marks: (don't write anything below)

1	2	3	4	Total
10	30	5	5	50

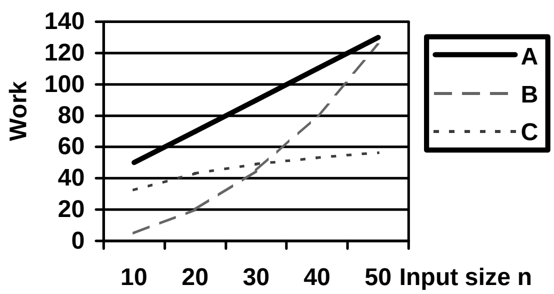
Part I (Multiple choice) Circle the letter of the best choice (only one)

1. (2 marks) Which one of these properties is NOT necessarily true for an algorithm:
    a. the order in which the steps are executed is well defined
    b. <u>it has at least one Get statement</u>
    c. each step is unambiguous
    d. each step is effectively computable
    e. it stops

2. (2 marks) How many name comparisons, in the worst case, does it take for a *sequential search* to locate a name in a list of *n* names?
    a. 1
    b. n/2
    c. <u>n</u>
    d. 2n
    e. n^2

3. (2 marks) For how many of the names does *binary search* require more than two comparisons to locate in the list: Ann, Bob, Garry, Jill, Kim, Mary, Sid
    a. 1
    b. 2
    c. 3
    d. <u>4</u>
    e. 5

4. (2 marks) The following graph demonstrates the relations between the input size and the amount of work done by 3 different algorithms: A, B, and C.



What order of magnitude would you expect algorithm A to be?
    a. $O(\log_2(n))$
    b. <u>O(n)</u>
    c. $O(n^2)$
    d. $O(2^n)$
    e. none of the above

5. (2 marks) The main reason why computers use the binary numbering system is:
    a. electrical devices cannot represent more than 2 digits
    b. they run faster when using binary numbers
    c. <u>reliability</u>
    d. a and c above
    e. none of the above

Part II: Algorithms

3. (4 marks) Given the following algorithm:

Get value for *n*
Set value of *x* to 0
Repeat until *n* < *1*
    Add n to x
    Print the value of *n* and *x*
    Decrease *n* by 1
End loop
Print the value of *n* and *x*
Stop

a) What does the algorithm print out when the input n is 3?

3 3
2 5
1 6
0 6

b) What does the algorithm print out when the input n is 0?

0 0
-1 0

4. (8 marks) Use s*election sort* to order the following list of numbers into ascending order (the algorithm is given below): 40, 30, 10, 20

1. Get values for *n* and the *n* list items
2. Set the marker for the unsorted section at the end of the list
3. Repeat steps 4 through 6 until the unsorted section of the list has one element
4. Select the largest number in the unsorted section of the list
5. Exchange this number with the last number in the unsorted section of the list
6. Move the marker for the unsorted section forward (left) one position.
7. Stop

a) Show the list after each exchange:

INITIAL LIST:	**40, 30, 10, 20**
	20, 30, 10, 40
	20, 10, 30, 40
	10, 20, 30, 40

b) How many comparisons of the elements in the list did the algorithm do in total?

3+2+1 = 6

c) How many exchanges of the elements in the list did the algorithm do in total? **3**

3. (3 marks) Write an algorithm that reads in two numbers and prints out their product. For example, given the input values 2 and 3 the algorithm would output 6.

   Get values for a, b
   Set value of c to a * b
   Print value of c

4. (5 marks) Write an algorithm that reads in 5 numbers and prints out their average. Use a loop to read in the numbers and sum them up.

```
Set sum to 0
Set i to 1
Repeat until i > 5
    Get value for n
    Add n to sum
    Add 1 to i
End loop
Set average to sum/5
Print average
```

5. (8 marks) Write an algorithm that goes through a list of 10 numbers and prints out an element if it is larger than the element following it in the list. For example, if given the input 1, 5, 2, 3 , 4, 1, 2, 6, 1 the algorithm would print out

   5
   4
   6

   You may assume that the numbers have already been read in by the algorithm and are stored in variables N_1, N_2, … N_{10}. Use a loop.

   Set i to 1
   While i < 10 do
       If $N_i > N_{i+1}$ then
         Print N_i
       Add 1 to i
   End loop

6. (2 marks) An algorithm that is $O(n^2)$ takes 10 seconds to execute on a particular computer when n = 200. How long would you expect it to take when n = 400? Briefly explain the reason for your answer.

   40 seconds because when we double the size of the input we expect the amount of work the algorithm does to increase 2^2 = 4 times. Note: different answers are also possible given consistent explanation.

Part III: Data Representation

1. (1 mark) Convert the binary number 11001 to decimal. Show your work.

   16 + 8 + 1 = 25

2. (2 marks) Convert the decimal number 21 to a binary notation. Show your work.

   16 8 4 2 1
   1 0 1 0 1 = 10101

3. (1 mark) Show how an 8-bit computer using sign/magnitude notation to represent negative integers stores the decimal value –21.

1	**0**	**0**	**1**	**0**	**1**	**0**	**1**

4. (1 mark) Use the ASCII table supplied on the next page to help you encode the following text as binary:

   D o g = **01000100 01101111 01100111**

Part IV: Boolean Logic and Gates.

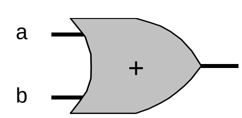
1. (3 marks) Are these Boolean expressions True or False, when X=10, Y=15, Z=20?

   a) (Z>5) AND (NOT (Y>X)) __**F**__ (T/F)

   b) NOT ((X<Y) OR (Y<Z)) AND (Y<Z) __**F**__ (T/F)

   c) (X=10) OR (X>Y) OR (Y=15) __**T**__ (T/F)

2. (2 marks) Following is a picture of a logic gate:



   a) What type of gate is this? **OR gate**

   b) Fill in the truth table describing the output of the gate:

a	b	Output
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

ASCII TABLE

Keyboard Character	Binary ASCII Code	Integer Equivalent	Keyboard Character	Binary ASCII Code	Integer Equivalent
(blank)	00100000	32	P	01010000	80
!	00100001	33	Q	01010001	81
"	00100010	34	R	01010010	82
#	00100011	35	S	01010011	83
$	00100100	36	T	01010100	84
%	00100101	37	U	01010101	85
&	00100110	38	V	01010110	86
'	00100111	39	W	01010111	87
(	00101000	40	X	01011000	88
)	00101001	41	Y	01011001	89
*	00101010	42	Z	01011010	90
+	00101011	43	[	01011011	91
,	00101100	44	\	01011100	92
–	00101101	45	]	01011101	93
.	00101110	46	^	01011110	94
/	00101111	47	_	01011111	95
0	00110000	48	`	01100000	96
1	00110001	49	a	01100001	97
2	00110010	50	b	01100010	98
3	00110011	51	c	01100011	99
4	00110100	52	d	01100100	100
5	00110101	53	e	01100101	101
6	00110110	54	f	01100110	102
7	00110111	55	g	01100111	103
8	00111000	56	h	01101000	104
9	00111001	57	i	01101001	105
:	00111010	58	j	01101010	106
;	00111011	59	k	01101011	107
<	00111100	60	l	01101100	108
=	00111101	61	m	01101101	109
>	00111110	62	n	01101110	110
?	00111111	63	o	01101111	111
@	01000000	64	p	01110000	112
A	01000001	65	q	01110001	113
B	01000010	66	r	01110010	114
C	01000011	67	s	01110011	115
D	01000100	68	t	01110100	116
E	01000101	69	u	01110101	117
F	01000110	70	v	01110110	118
G	01000111	71	w	01110111	119
H	01001000	72	x	01111000	120
I	01001001	73	y	01111001	121
J	01001010	74	z	01111010	122
K	01001011	75	{	01111011	123
L	01001100	76	:	01111100	124
M	01001101	77	}	01111101	125
N	01001110	78	~	01111110	126
O	01001111	79			