CMPUT 101- Solutions
Final Exam (2 hours)
August 17th, 2001

Last Name: ______________________________

First Name: ______________________________

Student ID: ______________

Signature: ______________________________

Section: B1

Instructor: I. E. Leonard

Instructions: Read carefully before proceeding.

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

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

Have a great summer!

Marks: (don't write anything below)

I	II	III	IV	V	VI	Total
20	13	14	10	7	36	100

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

1. (2 mark) 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. it has at least one Get statement
   c. each step is unambiguous
   d. each step is effectively computable
   e. it stops

2. (2 marks) Two algorithms A and B are given, of order O(n) and O(n^2) respectively. What does that say us about the algorithms?

   a. given input of size 10, algorithm A will perform 10 steps
   b. given input of size 10, algorithm B will perform 100 steps
   c. algorithm B always performs more work than algorithm A
   d. all of the above
   e. none of the above

3. (2 marks) The ASCII code mapping uses how many bits to represent each character:

   a. 1
   b. 3
   c. 8
   d. 16
   e. none of the above

4. (2 marks) A multiplexor with 4 input lines has how many selector lines:

   a. 1
   b. 2
   c. $2^4 – 1$
   d. 2^4
   e. none of the above

5. (2 marks) The term *address-space* refers to the:

   a. computer's maximum memory size
   b. width of each memory cell
   c. address of a memory cell storing the space character
   d. computer's hard-drive storage capacity
   e. none of the above

6. (2 marks) In a computer that has 64MB of memory, at a minimum, how many bits needs the MAR to be so all the memory cells can be addressed (hint: 2^{20} = 1MB)

   a. 20
   b. 24
   c. 25
   d. 26
   e. more than 26

7. (2 marks) A hard-drive has 10 sectors per track and a rotation speed of 6000 rev/min. The transfer time of this hard-drive is:

   a. 1 sec.
   b. 100 msec.
   c. 10 msec.
   d. 1 msec.
   e. none of the above

8. (2 marks) The program counter (PC)

   a. holds the next instruction to be executed
   b. holds the address of the next instruction to be executed
   c. counts how many programs are running
   d. counts how many times a program is executed
   e. is found in the ALU

9. (2 marks) Memory access time in today's computers is typically

   a. 1-2 nanoseconds
   b. 1-2 milliseconds
   c. 1-2 seconds
   d. 50-75 nanoseconds
   e. 50-75 milliseconds

10. (2 marks) Which of the following is NOT a responsibility of the operating system:

    a. user interface management
    b. program scheduling and activation
    c. translation of assembly into machine code
    d. efficient resource allocation
    e. deadlock detection, error detection

Part II: Algorithms

1. (4 marks) Given the following algorithm:

```
Get a value for N
Set the value of Result to 1
Repeat until N ≤ 0
    Set the value of Result to (Result x N)
    Decrease N by 1
End of loop
Print the value of Result and N
Stop
```

a. What does the algorithm print given the input value 4?

24 0

b. What does the algorithm print given the input value 0?

0 -1

2. (5 marks) Use *selection sort* to order the following list of numbers in an ascending order (the algorithm is given below): 4, 3, 1, 5, 2

```
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 of elements:

4, 3, 1, 5, 2 (initial)
4, 3, 1, 2, 5
2, 3, 1, 4, 5
2, 1, 3, 4, 5
1, 2, 3, 4, 5

b. How many comparisons of elements are made by the algorithm in total?

4 + 3 + 2 + 1 = 10

c. How many exchanges of elements are made by the algorithm in total?

4

2. (4 marks) The following list of names is given:

   Ann, Bob, Garry, Greg, Kim, Mary, Matt, Sid, Wong

   If using a *sequential search*:

   a. How many name comparisons are needed to locate the name *Kim* in the list?

      5

   b. Which of the names will take the most number of name comparisons to locate?

      Wong

   If using a *binary search*:

   c. How many name comparisons are needed to locate the name *Sid* in the list?

      3

   d. Which of the names will take the most number of name comparisons to locate?

      Greg, Wong

Part III: Binary Numbers and Circuits

1. (2 marks) Convert the numbers below to *decimal* (base-10):

   a. 1100 *binary* (base-2)

      $1 \times 2^3 + 1 \times 2^2 = 8 + 4 = 12$

   b. 23 *hexadecimal* (base-16)

      $2 \times 16^1 + 3 \times 16^0 = 32 + 3 = 35$

2. (4 marks) Show how the decimal number $-\frac{1}{16}$ is stored in a computer that uses 16 bits to represent real numbers (10 for the mantissa and 6 for the exponent, both including the sign bit). Show your work as indicated below.

   a. Show the binary representation of the number:

      - 0.0001

   b. Show the binary number in normalized scientific notation:

      -0.1×2^{-3}

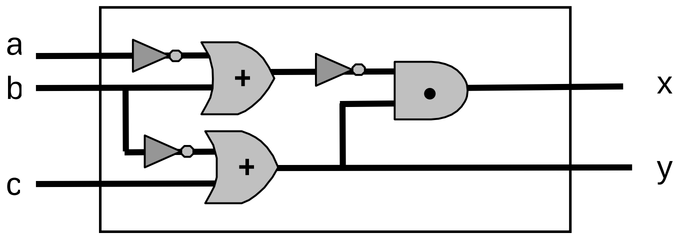
   c. Show how the binary number will be stored in the 16 bits below

1	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1

3. (1 mark) Is the following Boolean expression *true* or *false*, when A=2, B=4, C=6?

   (A>1) AND ((B>C) OR (A<B)) __T__ (T/F)

4. (2 marks) The following circuit is given:



For the two rows shown in the truth-table below, fill in the correct output values for *x* and *y*.

Input			Output	
a	**b**	**c**	**x**	**y**
1	0	0	**1**	**1**
1	1	0	**0**	**0**

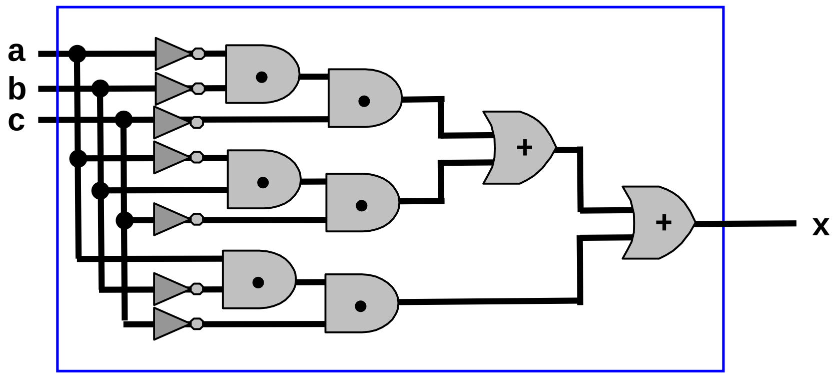
5. (5 marks) Use the ***sum-of-products*** circuit construction algorithm to design a circuit using AND, OR, and NOT gates that implements the following truth table.

Input			Output
a	**b**	**c**	**x**
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

a. Write the expression that describes the output

x = $\bar{a} \cdot \bar{b} \cdot \bar{c} + \bar{a} \cdot b \cdot \bar{c} + a \cdot \bar{b} \cdot \bar{c}$

b. Draw the circuit diagram.



Part IV: Von Neumann Architecture

1. (6 marks) Fill in the blanks.

The **_CCR (Condition Code Register)_** is a special purpose register whose bits are set as a result of comparing two values; it is found in the **______ALU_______** sub-system.

The main computer memory is often referred to as **______RAM_______**, whereas a memory that can only be read from but not written to is called **______ROM_______**.

A direct access storage device that has one read/write head per track is sometimes referred to as a **____drum________**. Compared to a regular hard-drive, it reduces the disk access time by eliminating the **_______seek time________**.

2. (4 marks) To ensure that the processor does not sit idle if there is useful work to do, the operating system keeps track of programs that are running on the computer by maintaining three lists: *Running*, *Ready*, and *Waiting*.
Briefly explain the role of the following two lists:

Running:

The program that is currently being executed by the CPU is kept the *Running* list (only one program can be in the *Running* list at any given time). Once the program's time-slice is up or it needs to do an I/O (or needs some other resource) it is moved to the *Ready* or *Waiting* list, respectively.

Waiting:

Programs that are waiting for an I/O (or some other resource) operation to finish are kept in the *Waiting* list. Once the operation is finished, a program is moved to the *Ready* list.

Part V: Assembly language

1. (3 marks) Assuming that memory locations X and Y store the values 5 and 3 respectively, what values will they and the register R store after the following assembly program fragment is executed?

	LOAD	X
	SUBTRACT	Y
	STORE	Y
	INCREMENT	X

Answer: X = __**6**__ Y = __**2**__ R = __**2**__

2. (4 marks) What does the following assembly program output when executed?

	.BEGIN	
	CLEAR	A
	CLEAR	C
Top:	LOAD	A
	ADD	Two
	STORE	A
	INCREMENT	C
	LOAD	Two
	COMPARE	C
	JUMPLT	Top
	JUMPEQ	Top
	OUT	A
	OUT	C
	HALT	
A:	.DATA	0
C:	.DATA	0
Two:	.DATA	2
	.END	

Answer: __**6**__ __**3**__

Assembly Language Instructions

INSTRUCTION		MEANING
LOAD	X	CON(X) → R
STORE	X	R → CON(X)
CLEAR	X	0 → CON(X)
ADD	X	R + CON(X) → R
INCREMENT	X	CON(X) + 1 → CON(X)
SUBTRACT	X	R – CON(X) → R
DECREMENT	X	CON(X) – 1 → CON(X)
COMPARE	X	If CON(X) > R then GT = 1 (EQ=0, LT=0) If CON(X) = R then EQ = 1 (GT=0, LT=0) If CON(X) < R then LT = 1 (GT=0, EQ=0)
JUMP	X	Transfer to memory location X
JUMPGT	X	Transfer to location X if GT = 1
JUMPEQ	X	Transfer to location X if EQ = 1
JUMPLT	X	Transfer to location X if LT = 1
JUMPNEQ	X	Transfer to location X if EQ = 0
IN	X	Input an integer value from the standard input device and store it in memory cell X.
OUT	X	Output, in decimal notation, the value stored in memory cell X.
HALT		Stop program execution.

PART VI: C++ programming

1. (4 marks) Name two things that are wrong with the following C++ program:

```
// This program reads in an integer and then displays it 5 times.
#include <iostream.h>
void main()
{
   const int count = 5;
   int n;

   cout << "Enter a number:";
   cin >> n >> endl;

   while ( count > 0  ) {
       cout << n << endl;
       count = count – 1;
   }
}
```

Answer:

1. *count* cannot be declared as a constant because *count* is later changed.
2. *endl* cannot be used with a cin statement, the statement should read *cin* >> *n*;

2. (6 marks) What do the following two programs output when executed?

a.

```
#include <iostream.h>
void main()
{
  int n, times;

  times = 0;
  n = 20;
  while ( n >= 5 )
  {
    ++times;
    n = n / 2;
  }
  cout << times << endl;
  cout << n << endl;
}
```

Answer: **3**
2

b.

```
#include <iostream.h>
void main()
{
  int i, pc, vc, P[3], V[3];

  P[0] = 10;
  P[1] = 20;
  P[2] = 30;
  i=0;
  while ( i < 3 ) {
    V[i] = 0;
     i = i + 1;
  }
  pc = 30;
  vc = 200;
  i = 0;
  while ( i < 3 ) {
     if ( pc == P[i] ) {
         V[i] = V[i] + vc;
     }
     i = i + 1;
  }
  cout << "i = " << i << endl;
  cout << V[0] << ", " << V[1]  << ", " << V[2];
}
```

Answer: **i = 3**
0, 0, 200

3. (4 marks) Write a C++ program that computes and displays the area of a triangle, after the user has entered its height and base. Hint: $area = (height \times base) / 2$

```
#include <iostream.h>
void main()
{
  double area, height, base;

  cout << "Enter height of triangle: ";
  cin >> height;
  cout << "Enter base of triangle: ";
  cin >> base;
  area = (height * base) / 2;
  cout << "Area of triangle is " << area;
}
```

4. (8 marks) Implement the following algorithm in C++:

```
Get values for integers $A_1$, $A_2$, …, $A_{20}$
Set the value of largest so far to $A_1$
Set the value of location to 1
Set the value of i to 2
While $i \leq 20$ do
    If $A_i$ > largest so far then
        Set largest so far to $A_i$
        Set location to i
    Add 1 to the value of i
End of the loop
Print out the values of largest so far and location
Stop
```

```
#include <iostream.h>
void main()
{
    const int MAX = 20;
    int i, a, A[MAX], largest_so_far, location;

    i = 0;
    while ( i < MAX ) {
        cout << "Enter a value: ";
        cin >> a;
        A[i] = a;
        ++i;
    }
    largest_so_far = A[0];
    location = 0;
    i = 1;
    while ( i < MAX ) {
       if ( A[i] > largest_so_far ) {
          largest_so_far = A[i];
          location = i;
       }
      ++i;
    }
    cout << "Largest value is " << largest_so_far << " at location " << location+1 << endl;
}
```

5. (14 marks) Write a C++ program that prompts the user to enter student's grades (the grades are integer values in the range 1-9). The program then displays the student's: *grade point average* (all courses have the same weight), the *lowest grade*, and *how many courses* the student failed (number of grades less than 4). If the student failed all his/her courses a message "What a bummer!" is displayed at the very end. An input value less or equal to 0 indicates that there are no more grades to be entered. An example running session looks as follows:

```
Enter a grade: 8
Enter a grade: 7
Enter a grade: 3
Enter a grade: 5
Enter a grade: 3
Enter a grade: -1

GPA: 5.2
Lowest grade: 3
Number of courses failed: 2
```

```
#include <iostream.h>
void main()
{
  int grade, count, lowest, failed;
  double sum, GPA;

  count = 0;
  sum = 0;
  lowest = 10;
  failed = 0;

  cout << "Enter a grade:";
  cin >> grade;
  while ( grade > 0 )
  {
    sum = sum + grade;
    if ( grade < lowest ) {
      lowest = grade;
    }
    if ( grade < 4 ) {
      ++failed;
    }
    ++count;
    cout << "Enter a grade:";
    cin >> grade;
  }
  GPA = sum / count;

  cout << endl;
  cout << "GPA: " << GPA << endl;
  cout << "Lowest grade: " << lowest << endl;
  cout << "Number of courses failed: " << failed << endl;
  if ( failed == count ) {
    cout << "What a bummer!" << endl;
  }
}
```