UNIVERSITY OF VICTORIA
EXAMINATIONS APRIL 2016

Course	CSC 111 Fundamentals of Programming with Engineering Applications
Section	201601
CRN	20678
Instructor	D. Michael Miller
Duration	Three (3) hours

NAME (PRINT}	
STUDENT NUMBER	V00
SIGNATURE	

THIS QUESTION PAPER HAS **10 PAGES** INCLUDING THIS TITLE PAGE.STUDENTS MUST COUNT THE NUMBER OF PAGES IN THIS EXAMINATION PAPER BEFORE BEGINNING TO WRITE, AND REPORT ANY DISCREPANCY IMMEDIATELY TO THE INVIGILATOR.

ANSWER ON THE EXAMINATION PAPER IN THE SPACES PROVIDED. YOU MAY NOT NEED ALL THE SPACE. USE THE BACKS OF PAGES FOR ROUGH WORK ONLY.

NO BOOKS OR NOTES ARE ALLOWED. YOU ARE PROVIDED ONE HANDOUT WITH THE C LANGUAGE OPERATORS AND SELECTED FUNCTIONS FROM <string.h>.

NO OTHER AIDS (E.G. CALCULATORS, LAP TOPS, TABLETS) ARE PERMITTED. ALL ELECTRONIC DEVICES INCLUDING CELL PHONES MUST BE OFF.

Question	*Max Marks*	*Score*
1	15	
2	10	
3	10	
4	10	
5	12	
6	10	
7	6	
8	8	
9	7	
10	12	
Total	*100*	

Question 1 [15] Circle the appropriate answer for each statement. Grading is +1 for each correct answer and -1/2 for each incorrect answer.

Each question is with reference to the C programming language.

A `long int` always occupies 64 bits in memory.	True False
Given that `x` is a `double`, what value is assigned to `x` by `x = 3.5 + 5 / 2;`	5 5.0 5.5 6.0 6
A *while* loop can always be rewritten as a *for* loop.	True False
Given `int x=10, y=12, z=15;`, the value of the expression `(x<y && y<z)` is	True False 0 1
A function must always return a value.	True False
A program is free to ignore the value returned by a function.	True False
A *do while* loop when reached may in some circumstances execute 0 times.	True False
A function can have a pointer to a function as a parameter.	True False
Once a pointer is assigned a value, that value cannot be changed.	True False
You can use `sizeof` to determine the size in bytes of any variable.	True False
Arrays of integers are NULL terminated.	True False
The function `feof` can be used to detect the end of each input line from `stdin`.	True False
If `ch` is a variable of type `char`, the following code will convert an uppercase letter to the corresponding lowercase letter: `if(ch>='A' && ch<='Z') ch+='a'-'A';`	True False
The function `malloc` can be used to allocate memory as needed while a program is running.	True False
If you pass an array as a parameter to a function, a copy of the array is made for use inside the function.	True False
Score	____ - 0.5 * ____ = _____ correct wrong

Question 2 [10] Answer each question in the space provided.

(a) What is the purpose of the line `#include <math.h>` that appears at the beginning of many C programs?

(b) Given three integer variables `a,b,c` whose values are the lengths of the sides of a triangle, write a C statement (or statements) that will assign an integer variable `isos` the value 1 if the triangle is isosceles (2 equal length sides) and 0 otherwise.

(c) Write a C statement (or statements) specifying a user-defined type for a point in 3-dimensional space where each value is represented by a `double`.

(d) Given that `q` and `sum` are `int` variables, describe what the following code does (you do not have to give an actual numeric result) **or** explain if you think there is an error.

```
for(q = 2, sum = 0; q <= 100; q += 2)
  sum += q;
```

(e) Explain the difference between `p` and `q` in the declaration `int  p, *q;`

Question 3 [10] Write a complete C program that will read an arbitrary number of integers from *stdin* using any nonnumeric (e.g. quit) to terminate the input. Your program is to compute and print the standard deviation of the values which is given by the formula

$$s = \sqrt{\frac{1}{n-1}\sum_{i=0}^{n-1} x_i^2 - \frac{(\sum_{i=0}^{n-1} x_i)^2}{n}}$$

Where the x_i are the values read and n is the number of values read (assume that at least two values are read, *i.e.* $n>1$). **NOTE an array is not required.**

Question 4 [10] Write a complete C program that will read up to 100 `double` values using any nonnumeric to terminate the input. Your program is to compute and print the average value. It is to then print the input values (one per line) in the order they were read that are greater than or equal to the average.

Question 5 [12] Write a C function called `ordered` that has two parameters: `int arr[]` and `int n` where `n` is the number of values in the array. Your function is to return an `int` equal to `1` if the values in `arr` are in strictly ascending order, `-1` if the values in `arr` are in strictly descending order, and `0` if they are not in either order. You can assume there are at least two different values in the array. **NOTE**: Your function is to check order, it is not to do a sort.

Question 6 [10] Complete the following program so that it behaves as described in the comments. **Look through the complete program before starting to fill-in your answers.**

```
#include <stdio.h>

#include <stdlib.h>

int main(){

  char fileName[256], ch;

  FILE *fin;

  ____________________;                    // define variables

  ____________________;

  printf("Enter file name: ");

  scanf("%s",______________);              // read input file name

  fin = fopen(______________,______);      // open the input file

  if(____________){

    printf("Could not open the file\n");

    return 1;

  }

  while(1){

    fscanf(fin,"____",____);           // read one integer value

    if(____________) break;            // break if at end of file

    sum+=x;

    cnt++;

  }

  printf("Number of values: %d\n",cnt);

  if(cnt>0)

    printf("Average: %f\n",____________);  // display average

  return EXIT_SUCCESS;

}
```

Question 7 [6] Consider the following type definition:

```
typedef struct{
  char name[64];
  double mark;
} student;
```

Complete the following function so that it performs the operation described in the comments. You may not need all the space provided.

```
void sortList(student list[],int n){
// Sort a list of n students into ascending
// order by mark.
// Students with equal marks are ordered by
// ascending name.
  student t;
  int i,j,maxp;
  for(i=0;i<n-1;i++){
    maxp = i;
    for(j=i+1;j<n;j++)
      if(____________________________________________
         ____________________________________________
         ____________________________________________
      ) maxp=j;
    if(maxp != i){
      ______________________________
      ______________________________
      ______________________________
      ______________________________
    }
  }
  return;
```

Question 8 [8] A matrix is square if it has the same number of rows and columns. The transpose of a square matrix is found by exchanging the value in row *i* and column *j* with the value in row *j* and column *i* for all rows and columns.

Given the following header, complete the function so that it transposes the matrix given by `a` which has `d` rows and `d` columns where `d <= 100`.

```
void transpose(double a[100][100], int d){
```

Question 9 [7] Ackermann's function $A(m,n)$ where m and n are integers ≥0 is defined by

$$A(m,n) = \begin{cases} n+1 & \text{if } m=0 \\ A(m-1,1) & \text{if } m>0 \text{ and } n=0 \\ A(m-1, A(m,n-1)) & \text{if } m>0 \text{ and } n>0 \end{cases}$$

Write a C function named `A` to compute Ackermann's function using recursion.

Question 10 [12] Consider the following typedefs:

```
typedef struct nodeS *nodePtr;
typedef struct nodeS{
  double x;
  nodePtr next;
} nodeT;
```

Write a function with the header `double avgList(nodePtr list)` that traverses a linked list built using the above typedefs and returns the average of the values in the list. You can assume the list is not empty.

```
double avgList(nodePtr list)
```

****** *End of Exam* ******

CSC111

Operator	Description	Associativity
++ --	Postfix increment and decrement	Left-to-right
()	Function call (see note 1)	
[]	Array subscripting	
.	Element selection by reference	
->	Element selection through pointer	
++ --	Prefix increment and decrement	Right-to-left
+ -	Unary plus and minus	
! ~	Logical NOT and bitwise NOT	
(*type*)	Type cast	
*	Indirection (dereference)	
&	Address-of	
sizeof	Size-of	

Operator	Description	Associativity
* / %	Multiplication, division, modulus (remainder)	Left-to-right
+ -	Addition and subtraction	
<< >>	Bitwise left shift and right shift	
< <=	Relational "less than" and "less than or equal to"	
> >=	Relational "greater than" and "greater than or equal to"	
== !=	Relational "equal to" and "not equal to"	
&	Bitwise AND	
^	Bitwise XOR (exclusive or)	
|	Bitwise. OR (inclusive or)	
&&	Logical AND	
||	Logical OR	
? :	Ternary conditional	Right-to-left
=	Assignment	
+= -=	Assignment by sum, difference	
*= /= %=	Assignment by product, quotient, remainder	
<<= >>=	Assignment by bitwise left shift, right shift	
&= ^= |=	Assignment by bitwise AND, XOR, OR	
,	Comma	Left-to-right

NOTE 1: Brackets are used to override the default precedence.

Selected functions from <string.h>

		(limited to n characters)
Concatenation	char *strcat(char * dest, const char * src);	char *strncat(char * dest, const char * src, size_t n)[
First occurrence of a character	char *strchr(const char * src, int c);	
String comparison	int strcmp(const char * src1, const char * src2);	int strncmp(const char * src1, const char * src2, size_t n);
String copy	char *strcpy(char * dest, const char * src);	char *strncpy(char * dest, const char * src, size_t n);
String length	size_t strlen(const char * src);	
Find a substring	char *strstr(const char *haystack, const char *needle);	