# UNIVERSITY OF VICTORIA
# EXAMINATIONS APRIL 2017

| | |
|---|---|
| Course | CSC 111 Fundamentals of Programming with Engineering Applications |
| Section | A01 |
| CRN | 20691 |
| Instructor | D. Michael Miller |
| Duration | Three (3) hours |

| | |
|---|---|
| NAME (PRINT} | |
| STUDENT NUMBER | V |
| SIGNATURE | |

THIS QUESTION PAPER HAS **9 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 | 8 | |
| 7 | 8 | |
| 8 | 15 | |
| 9 | 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 double` always occupies 8 bytes in memory. | True False |
| Given that `x` is a `double`, what value is assigned to `x` by `x = 3.5 + 3 / 2;` | 3 3.25 3.5 3.75 4.0 4.25 4.5 other |
| A *while* loop can always be rewritten as a *for* loop. | True False |
| Given `int x=10, y=15, z=12;`, 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 *for* loop must always execute at least once when encountered during execution of a program. | True False |
| A function parameter can be a pointer to a function. | True False |
| Consider `int *p;` If `p` has value hexadecimal `600000`, what value will it have after executing `p++;` | 600000 600001 600004 other |
| You can use `sizeof` to determine the size in bytes of any type. | True False |
| The function call `strcmp("apples","apple")` will return a value that is | negative zero positive |
| The statement `double *p[100];` will produce a compiler syntax error. | True False |
| If `ch` is a variable of type `char`, the following code will convert a lowercase letter to the corresponding uppercase letter: `if(ch>='a' && ch<='z') ch+='a'-'A';` | True False |
| The functions `calloc` and `realloc` can both be used to allocate memory 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 <stdio.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 `equ` the value 1 if the triangle is equilateral (3 equal length sides) and 0 otherwise.

(c) Write a C statement (or statements) specifying a user-defined type (called `time`) with components `hour` and `minutes` as `int`, and component `seconds` as 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 = 1, sum = 0; q < 100; q += 2)
  sum += q*q;
```

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

**Question 3 [10]** Write a complete C program that will input 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 *range* of the values which is the maximum value read minus the minimum value read. Note that the range can be 0 which is the case where the minimum equals the maximum, i.e. all input values are equal. **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 input that are within 10% of the average.

**Question 5 [12]** Write a C function called `check` 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 all $\geq 0$ ,
`-1` if the values in `arr` are all $< 0$, and
`0` if there is a mix of the two conditions.

```
int check(int arr[], int n)
```

**Question 6 [8]** Write a C function called `shuffle` with the header shown in the box below. Your function is to fill the array `out` by alternately taking strings in order from `x` and `y`, i.e. the first four values in `out` will be `x[0] y[0] x[1] y[1]`. Arrays `x` and `y` both contain `n` strings and you can assume `out` is big enough to hold `2*n` strings.

```
void shuffle(char *out[], char *x[], char *y[], int n)
```

**Question 7 [8]** Complete the following function so that it performs a binary search recursively to find the **position** of the value of `key` in `list`. The values in `list` are in ascending order and there are no duplicate values. `start` and `end` are the first and last positions for the range being searched. The function is to return `-1` if the `key` value is not found.

```
int search(int list[],int start, int end, int key){
    int m;
    if(end<start) _____;
    m= _____;
    if(key==list[m]) _____;
    else if(key<list[m]) return _____;
    else return _____;
}
```

Given
```
int x[]={2, 4, 5, 8, 9, 13, 16, 23};
int n=sizeof(x)/sizeof(double);
```

What will be displayed by `printf("search returns: %d\n", search(x,0,n-1,13));`

Answer: _____

**Question 8 [15]** Complete the following program so that it behaves as described in the comments and as shown in the sample output. Look through the complete program before starting to fill-in your answers.

```
#include <stdio.h>
void sort(double x[],double y[],int n){
// sort pairs of values in arrays x and y by ascending value of x and
// for equal values of x by descending value of y
  int i, j, pos; double t;
  for(i=0;i<n-1;i++){
    for(pos=i,j=_____;_____;_____)   // find position (pos) for swap
      if(x[j]__x[pos] _____ x[j] == x[pos] ____ y[j]__y[pos]) _____;
    if(i!=pos){
      t=x[i]; _____; _____;    // swap x for positions i, pos
      _____; _____; _____; // swap y for positions i, pos
    }
  }
  return;
}
int main(){
  double x[100],y[100],a;
  int n=0,i;
  // read data with stopping value of -999
  while(1){
    scanf("%lf",&a);
    if(_____) break;
    x[n]=_____;
    scanf("%lf",_____);
  }
  printf("Data as read:\n");  // display data as read with one pair per line
  for(_____) printf("%f %f\n",x[i],y[i]);
  sort(x,y,n);                // do sort
  printf("Sorted data:\n");   // display sorted data with one pair per line
  for(_____) _____;
  return 0;
}
```

**Sample output**
**(input is in bold underlined)**

**1 2**
**2 3**
**3 4**
**1 0.5**
**2 2.5**
**-999**
Data as read:
1.000000 2.000000
2.000000 3.000000
3.000000 4.000000
1.000000 0.500000
2.000000 2.500000
Sorted data:
1.0 2.0
1.0 0.5
2.0 3.0
2.0 2.5
3.0 4.0

**Question 9 [12]** Consider the following typedef:

```
typedef struct{
  char name[50];
  char vnumber[10];
  int grade;
} student;
```

(a) Write a function named `inputRecord` that will input the information for one student assuming the input looks like this (the comma always immediately follows the surname):

```
V00099999 Turtle, Michelangelo 95
```

(b) Write a function named `outputRecord` that will accept a student record as a parameter and display it using `printf` in the same format as above.

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

CSC 111 Spring 2017

## SELECTED FUNCTIONS

| 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>**

| | |
|---|---|
| Concatenation | char *strcat(char * dest, const char * src); |
| String comparison | int strcmp(const char * src1, const char * src2); |
| String copy | char *strcpy(char * dest, const char * src); |
| String length | size_t strlen(const char * src); |