TCB3
209

THE UNIVERSITY OF THE WEST INDIES
ST. AUGUSTINE

EXAMINATIONS OF April/May 2019

Code and Name of Course: COMP1602 – Computer Programming II

Date and Time: Wednesday 8th May 2019 4 pm Duration: 2 Hours

INSTRUCTIONS TO CANDIDATES: This paper has 5 pages and 3 questions

Answer *ALL* Questions

Questions are not evenly weighted.

The use of programmable calculators is not allowed.

1) A C-string is a character array with a null terminating character ('\0'). In this question, you are not allowed to use any of the functions built into the C-string library except *strlen*.

   a) Write a function *toUpper* which given a character *ch* returns the upper case of *ch* if it is a letter. The function has the following prototype:

   ```
   char toUpper (char ch);
   ```

   [2 marks]

   b) **BRB** (**B**e **R**ight **B**ack) and **SMH** (**s**haking **m**y **h**ead) are some terms you may come across in e-mails, chat rooms, online games, instant messaging, or elsewhere on the Internet or in phone text messages. BRB and SMH are examples of *net lingua* (the language of the Internet) and are formed by combining the upper case of the first letter of each word.

   A C-string *s*, contains a message consisting of words separated by one or more spaces. The words may have both upper case and lower case letters. Write a function, *toNetLingua*, which takes two C-strings, *s* and *n*, as parameters and finds the net lingua of the message in *s* and stores it in *n*. The function has the following prototype:

   ```
   void toNetLingua (char s[], char n[]);
   ```

   [6 marks]

   c) Write a function, *intToString*, with the following prototype:

   ```
   void intToString(int n, char s[]);
   ```

   which accepts an integer value *n* and a C-string *s* as parameters and stores *n* as a sequence of characters in *s*.

   For example,

   If $n = 0$, *s* should contain '0'
   If $n = 90$, *s* should contain '9', '0'
   If $n = 125$, *s* should contain '1', '2', '5'

   [7 marks]

Total Marks: 15

2) A two dimensional (2-D) integer array A has n rows and m columns where $0 < n \leq 100$ and $0 < m \leq 100$. If n is equal to m, A is said to be *square*.

a) Write a function, *printRowReverse*, which given A, n, m and a row number *row* as parameters, prints the values of *row* in reverse order. The function has the following prototype:

```
void printRowReverse(int A[][100], int n, int m, int row);
```

[2 marks]

b) Write a function, *printSerpentine*, which given A, n, and m as parameters, prints the contents of A in a serpentine fashion. For example, if

$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ the ouput is 1 2 3 6 5 4 7 8 9

and if $A = \begin{bmatrix} 5 & 1 \\ 8 & 9 \\ 2 & 4 \\ 3 & 7 \end{bmatrix}$ the ouput is 5 1 9 8 2 4 7 3

The function has the following prototype:

```
void printSerpentine(int A[][100], int n, int m);
```

[4 marks]

c) Assume that A is square.

i) A is said to be *topsy-turvy* if all the values above the leading diagonal are larger than all the values below the leading diagonal. Write a function *isTopsyTurvy* which given A and n (the number of rows and columns in A) as parameters, returns *true* if A is topsy-turvy and *false*, otherwise.

The prototype of *isTopsyTurvy* is as follows:

```
bool isTopsyTurvy(int A[][100], int n);
```

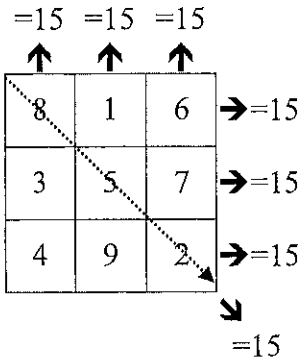
[5 marks]

Question 2 continues on the next page.

ii) *A* is called a *magic square* if the sum of the values in each row, column and the main diagonal is the same.

For example, the following 2-D array is a magic square since,

- The sum of each row is 15.
- The sum of each column is 15.
- The sum of the diagonal is 15.



Write a function, *isMagicSquare*, which given *A* and *n* (the number of rows and columns in *A*) as parameters, returns *true* if *A* is a magic square and *false*, otherwise.

The prototype of *isMagicSquare* is as follows:

```
bool isMagicSquare(int A[][100], int n);
```

[9 marks]

Total marks: 20

3) a) An integer array *A*, contains the following values:

	0	1	2	3	4	5	6	7	8	9	10
A	32	8	29	6	17	82	3	12	69	23	83

i) A *binary search* is used to search for the value 69 in the array *A*. Will the binary search find 69? Show the steps used in deriving your answer. [3 marks]

ii) If all the criteria for the binary search were met by the array *A*, how many comparisons will be made before 69 was found? Show your working. [4 marks]

iii) Suppose that a linear search is made on a **sorted** array. What statement can you make about the relative efficiency of the binary search algorithm compared to the linear search algorithm on a sorted array? [2 marks]

Question 3 continues on the next page.

b) The following array, A, needs to be sorted in ascending order:

	0	1	2	3	4	5
A	32	8	29	6	17	82

Starting with the contents of A as shown above, *draw* the modified array:

i) After **each** of the first three (3) passes of insertion sort.
ii) After **each** of the first three (3) passes of bubble sort.
iii) If the array was already sorted, would insertion sort or bubble sort be more efficient?

Show all working. [6 marks]

c) The ***mysterySort*** function below is a modified bubble sort. It is called with an array A and an integer n where n is the number of elements in A. The code for ***mysterySort*** is as follows:

```
void swap(int a[], int i , int j) {
    int temp = a[i];
    a[i] = a[j];
    a[j] = temp;
}

void mysterySort(int arr[], int n) {
    bool swapped = true;
    int j = 0;
    while (swapped) {
        swapped = false;
        j = j + 1;
        for (int i = j-1; i < n-1; i = i+1) {
            if (arr[i] > arr[i + 1]) {
                swap(arr, i, i+1);
                swapped = true;
            }
        }
        swap(arr, j-1, n-1);
    }
}
```

The array A is shown below.

A	37	8	29	6

For each of the passes of the outer *while* loop of *mysterySort,* show the contents of the array A. [5 marks]

Total Marks: 20

Total marks: 55
End of Examination