National University of Computer and Emerging Sciences

School of Computing **Fall 2017** **Islamabad Campus**

CS-201 Data Structures

Serial No:

Sessional-I

Total Time: 1 Hour

Total Marks: 60

Monday, September 18, 2017

Course Instructors

Dr. M. Adnan Tariq

Mr. Syed Muhammad Hassan Mustafa

Mr. Zeeshan Waheen

_____

Signature of Invigilator

_____	_____	_____	_____
Student Name	Roll No	Section	Signature

DO NOT OPEN THE QUESTION BOOK OR START UNTIL INSTRUCTED.

Instructions:

1. Attempt on question paper. Attempt all questions. Read the question carefully, understand the question, and then attempt it.
2. No additional sheet will be provided for rough work. Use back side of the paper for rough work.
3. After asked to commence the exam, please verify that you have Seven (x) different printed pages including this title page. There are a total of x questions.
4. Calculator sharing is strictly prohibited.
5. Use permanent ink pens only. Any part done using soft pencil will not be marked and cannot be claimed for rechecking.
6. Do not use Red color for drawing.

	Q-1	Q-2	Q-3	Q-4	Q-5	Q-6	Total
Marks Obtained							
Total Marks	**7**	**8**	**10**	**10**	**10**	**15**	**60**

Vetted By: _____ **Vetter Signature:** _____

Question 1 [7 Marks]

Consider the following array declaration in C++:

```
float Table [20][50]
```

Suppose the base address of `Table` is 500 (in decimal) stored in column major order.

Find the address (in decimal) of `Table [10][5]` (show all steps).

i) Address of the Required Column [4 marks]:

ii) Address of the Required Element [3 marks]:

Question 2 [8 Marks]

Consider the following code and provide answers to the following questions.

```
template <class Item>
void Sort(Item a[], int n)
{
   for (int i = 1; i < n; i++)// outer loop
   {
      Item tmp = a[i];
      for (int j=i; j>0 && tmp > a[j-1]; j--)
         a[j] = a[j-1];
      a[j] = tmp;
   }
}
```

i) Provide name of this sorting algorithm [1 mark]:

ii) Perform dry run on this algorithm using following data array and update this array after each iteration of the outer loop. [5 marks]

n=6

100	0	55	90	70	205

i=1

i=2

i=3

i=4

i=5

iii) Number of comparisons in the worst case [1 mark]:

iv) Number of swaps in the worst case [1 mark]:

Question 3 [10 Marks]

Consider the following array implementation of list and provide answers to the following questions:

Use updated list for each step:

I	Data [I]	Next [I]
0	7	5
1	10	0
2	*	*
3	80	4
4	9	-1
5	30	3

i) Insert '50' before '10' and provide updated list (order should remain same).

I	Data [I]	Next [I]
0		
1		
2		
3		
4		
5		

ii) Delete '7' and provide updated list (order should remain same).

I	Data [I]	Next [I]
0		
1		
2		
3		
4		
5		

iii) Insert ‘11’ after ‘30’ and provide updated list (order should remain same).

I	Data [I]	Next [I]
0		
1		
2		
3		
4		
5		

iv) Delete “9” and provide updated list (order should remain same).

I	Data [I]	Next [I]
0		
1		
2		
3		
4		
5		

Question 4 [10 Marks]

Consider the following bubble sort algorithm. This algorithm takes an integer array as an input and sorts the elements of that input array in ascending order (i.e., in increasing order of their values). There is no error or bug in the function.

```
void BubbleSort( int array[])
{
1    int pass = 1;
2   boolean exchanges;
3   do {
4        exchanges = false;
5        for (int i = 0; i < ARRAY_SIZE-pass; i++){
6             if (array[i] > array[i+1]) {
7                  int tmp = array[i];
8                  array[i] = array[i+1];
9                  array[i+1] = tmp;
10                 exchanges = true;
11            }
12       }
13       pass++;
14  } while (exchanges);
}
```

Modify the bubble sort algorithm to sort the (singly) linked list of integer values in descending order. For your convenience the declaration of the `class Node` and the structure of the bubble sort algorithm is given below (see next page). The structure of the bubble sort algorithm depicts line numbers. The statements (i.e., code) for some lines and while/if conditions are missing.

You have to write C/C++ code for empty lines as well as missing while (on line 6) and if (on line 8) conditions. The code must be written in the provided spaces. It is important to note that apart from empty lines no additional code is required to implement the desired functionality. [1 mark for each empty line and missing while/if condition]

```
Class Node {
     int val;
     Node* next;
};

Void BubbleSort( Node* head)
{
1    int pass = 1;
2    boolean exchanges;

3

4    do {
5         exchanges = false;
6         while (                                          ){


7

8              if (                                      ) {


9

10

11

12


13             } // end if statement

14


15        } // end while statement
16        pass++;
17   } while (exchanges); // end do while statement
}
```

Question 5 [10 Marks]

One important task of operating system (OS) is to schedule CPU processing time to run different processes. At a given time many different processes want to use CPU. It is the task of OS to select one process and send it to the CPU for execution. One of the scheduling techniques used by OS is called Round Robin. In this technique, currently running processes are executed for a fixed time slice one by one. The processes are always executed in the same order – unless a process is terminated or a new process is started by the user. For instance, if the time slice is 2ms and three processes P1, P2 and P3 are currently running in the system, the round robin strategy will execute processes in the following order.

P1 → P2 → P3 → P1 → P2 → P3 → P1 ….. (goes on until processes are finished)

Different data structures can be used by operating system to keep track which process to be executed next (once the time slice of the currently running process is finished). Typically such a data structure stores identifier of the process (i.e., process ID) along with other required data members (e.g., pointers in the case of linked list).

In the following, write code for selecting processes in a round robin manner using two different data structures, i.e., i) circular linked list and ii) queues (pointer-based implementation). In particular, you have to write the code of the `getNextProcess` function which gets as input the information about the current process and returns the next process to be executed.

i) **Circular Linked List:** In the code below write the missing statement. The `getNextProcess` function takes as input the pointer to the node of the process currently being executed. For your convenience declaration of `class Node` is also given below. It is important to note that apart from the empty line no additional code is required to implement the desired functionality. [2 marks]

Using Circular Linked Lisk:

```
class Node{
     int process_id;
     Node* next;
};

Node* getNextProces(  Node* currProcess){

1:


}
```

ii) **Queues:** In the code below write the missing statements. The `getNextProcess` function takes as input the id of the process currently being executed. It is important to note that apart from the empty lines no additional code is required to implement the desired functionality. [4 marks]

Using Queues:

```
class QueueNode{
     int process_id;
     Node* next;
};

Queue q; //global variable of class Queue

int getNextProcess( int process_id){

1:


2:


}
```

iii) **Performance Comparison:** Which data structure (circular linked list or queues) is better with respect to the performance of `getNextProcess` function? Justify your answer. The justification must not be longer than two lines. [2 marks]

iv) **Performance of adding a new process:** If a new process P4 is started by user, this process P4 has to be entered in the data structure for round robin scheduling. Which data structure (circular linked list or queues) is better with respect to the performance of adding a new process? Justify your answer. The justification must not be longer than two lines. (Hint: In case of circular linked list the new process has to be added before the process currently being executed). [2 marks]