National University of Computer and Emerging Sciences

FAST School of Computing **Spring 2020** **Islamabad Campus**

CS-201/218:Data Structures

Remote Final Exam

Attempt Time: 3 Hours
Submission (on google classroom and through email) Time: 15 minutes

Thursday, June 25, 2020

Course Instructor(s)

Dr. Abdul Waheed & Mr. Shehreyar Rashid

Total Marks: 100

Instructions:

1. The final exam will be attempted offline in the student's own handwriting (in readable way).
2. The students will use A4 size blank white sheets to attempt the exam (portrait format unless a diagram or table requires landscape). Each sheet of the A4 size paper **MUST** have the Roll Number, Name, the course code, name of the course and Signature of the student at the top of **EACH** sheet.
3. Students will use cam-scanner, MS lens, or an equivalent application to scan and convert their hand-written answer sheets into a **SINGLE** pdf file (keeping the correct order of pages and question numbers), which they will submit on LMS and **MUST also** email to the email address (of the concerned course/lab instructor) which will be provided. They will be given 15 minutes (after the 3 hours attempt time) for this purpose. All students must use the standard file name format (Full course code - Roll number e.g. CS201/218-18i-0001). Submissions after 30 minutes may not be accepted. **Try to submit soon after 3 hours of attempt time and do not wait for 15 minutes to be elapsed.**
4. For proven cheating/ plagiarism, student will get an F grade even if the student had opted for S/U grade, and the case will be referred to DDC (Department's Disciplinary Committee). Instructors will conduct vivas of randomly selected students or in case of doubt (significantly different attempt as compared to past performance in the course or matching attempt with other students). Plagiarism includes sharing an attempt to other students (copy providing). Students who are not able to satisfactorily answer instructor's questions (based on the exam as well as slightly lateral but related concepts) during viva will also be considered as plagiarism cases.
5. Students should carry a clean scanning that is free from any marks/stains etc.

	Q-1	Q-2	Q-3	Q-4	Q-5	Q-6	Q-7	Q-8	Total
Marks	**15**	**15**	**17**	**10**	**8**	**10**	**15**	**10**	**100**

Question 1 [10 + 5 Marks]

1. Given the following input, perform sorting using the specified algorithm:

3	1	4	2	9	11	7	5	6

a. Optimized Bubble sort:

Before Sorting	**3**	**1**	**4**	**2**	**9**	**11**	**7**	**5**	**6**
Pass 01									

b. Insertion sort:

Before Sorting	**3**	**1**	**4**	**2**	**9**	**11**	**7**	**5**	**6**
Pass 01									

2. Consider the following declaration of a multi-dimensional array in C++:

int arr[5][3][4][7][3];

Assuming that the base address is 0x12348 (hexadecimal). Are these valid indexes? If yes, what addresses will be pointed by?

a. a[4][1][0][5][1]
b. a[1][0][4][3][1]

Question 2 [15 Marks]

Answer the following questions in max two (2) lines.

1. Given a sorted stack, what is the maximum number of elements that you require to visit in order to search an element? Justify [1 mark]

2. Given a binary tree, what is the maximum number of elements that you require to visit in order to search an element? Justify [1 mark]

3. Given a BST, what is the maximum number of elements that you require to visit in order to search an element? Justify [1 mark]

4. Given an AVL tree, what is the maximum number of elements that you require to visit in order to search an elements? Justify [1 mark]

5. Given the following pseudo-code, what problem **mystery** function solves: [1 mark]

```
1 void mystery(int n)
2 {
3       Stack S;
4       while (n > 0)
5       {
6               push(&S , n%2);
7               n = n / 2;
8       }
9       while (!isEmpty(&S))
10              cout << pop(&S);
11 }
```

6. A single array A[1..MAXSIZE] is used to implement two stacks. The two stacks grow from opposite ends of the array. Variables top1 and top2 (topl > top2) point to the location of the top most element in each of the stacks. If the space is to be used **efficiently**, the condition for "isStackFull" is: [2 mark]

7. A Priority-Queue is implemented as a Max-Heap. Initially, it has 5 elements. The level-order traversal of the heap is given as: 10, 8, 5, 3, 2. Two new elements "1" and "7" are inserted in the heap in that order. The level-order traversal of the heap after insertion of the two elements is? [2 Marks]

8. Consider the following implementation of binary search. [6 Marks]

```
1  int search(char array[], char key)
2  {
3      int low = 0;
4      int high = strlen(array) - 1;
5      while (low <= high)
6      {
7          int mid = (low + high) / 2;
8          if (key > array[mid])
9              low = mid + 1;
10         else if (array[mid] > key)
11             high = mid - 1;
12         else
13             return mid;
14     }
15 return -1;
16 }
```

Suppose we change the code in the following ways. For each change, state whether the code will still work as intended? If not, explain what can go wrong?

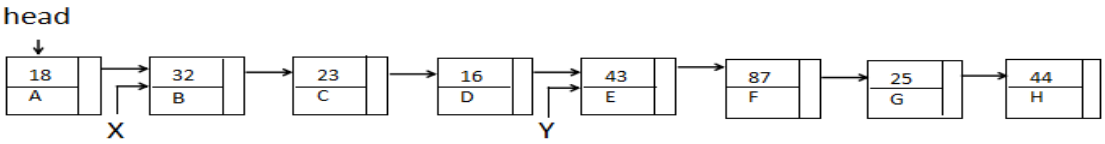
a. Change the line 5 to while (low < high)

b. Change line 7 to int mid = low;

c. Change line 9 to low = mid;

Question 3 [4 + 5 + 8 marks]

1. Consider the singly linked list with no (previous pointer) or tail as shown in the following figure where **X** (pointing to the 2nd node) and **Y (**pointing to the 5th node**)** are two pointers. For each question, assume the original list.



   a. Write down the structure of a node of the above list.[1 Mark]

   b. Write the shortest possible code to swap the info of nodes X and Y. [1 Mark]

   c. Write the generalized code to create and insert a new node before **head**. [2 Marks]

2. A student has been asked to write C++ code that reverses the singly link list using recursion. Student is very honest and has decided not to take help from any fellow students or internet and has written the code given as follows:

```
1 Node* reverse(Node* head)
2 {
3       if (head == NULL)
4             return NULL;
 5      if (head->next == NULL)
6             return head;
7       Node* nextNext = reverse(head->next);
8       nextNext->next=head;
9       head->next = NULL;

10      return head;
11 }
```

You can assume structure of Link List node as follows:

```
struct Node
{
    int value;
    Node* next;
};
```

And there is **NO SYNTAX ERROR**

Based on the code given, answer the following question:

Is this code working correctly? If yes, explain the logic (**do not explain the code line by line, just logic**). If code is not working correctly, identify the logical problem and correct the code (**do not rewrite complete code. Just specify line numbers along with modified code**). Remember yes/no will not get you marks. You have to explain in **max two lines**.

3. Write a **recursive algorithm** that inserts a node in singly Link List at given index**.** Prototype of function is: ***Node* InsertNode(Node* head, int val, int index)*** And assume structure of Link List node is :

```
struct Node
{
        int value;
        Node* next;
};
```

Insertion must be in such a way that it should cover all the cases (also provide main function). And for invalid index it should not affect the link list. Do not use static variable.

Question 4 [4 + 6 marks]

Some of you guys think that it is impossible to access any element in a given queue. However, a few still think that, given the primitive queue operations (enqueue, dequeue, front, size, IsFull and IsEmpty) only, we can access any element in a queue. This is a confusing situation among students, so we should give it a try to reach a consensus!

As a challenge, you are required to only use the primitive queue operations (enqueue, dequeue, front, size, IsFull and IsEmpty) to implement the methods requested.

You need to provide code for the following methods:

(i) **Get the nth element from the front of the queue, only removing the element requested (the nth element).**

Example: Get 4th element.
Initial State of Queue: 10 => 5 => 2 => 9 => 4 => 7
It will return 9 & Queue will be 10=> 5 => 2 => 4 => 7

(ii) **Get the nth element from the rear/back of the queue, removing only the required element.**

Example: Get 4th element.
Initial State of Queue: 10 => 5 => 2 => 9 => 4 => 7
It will return 2 & Queue will be 10=> 5 => 9 => 4 => 7

Note: **You are not allowed to use any other Data structure (e.g. Array, Link List OR Stack) or modify Queue operations**. Just use the Queue data structure.

Question 5 [Marks 8]

Convert the **Infix expression "**A / B * C – D + E / F / G - H**"** to **Prefix expression** using given algorithm. You need to just represent the required steps in a tabular form (as given in slides) and no C++ code is required.

Infix String: "A/B*C–D+E/F/G-H"
rev String:

symb	postfix	opstk
	…	

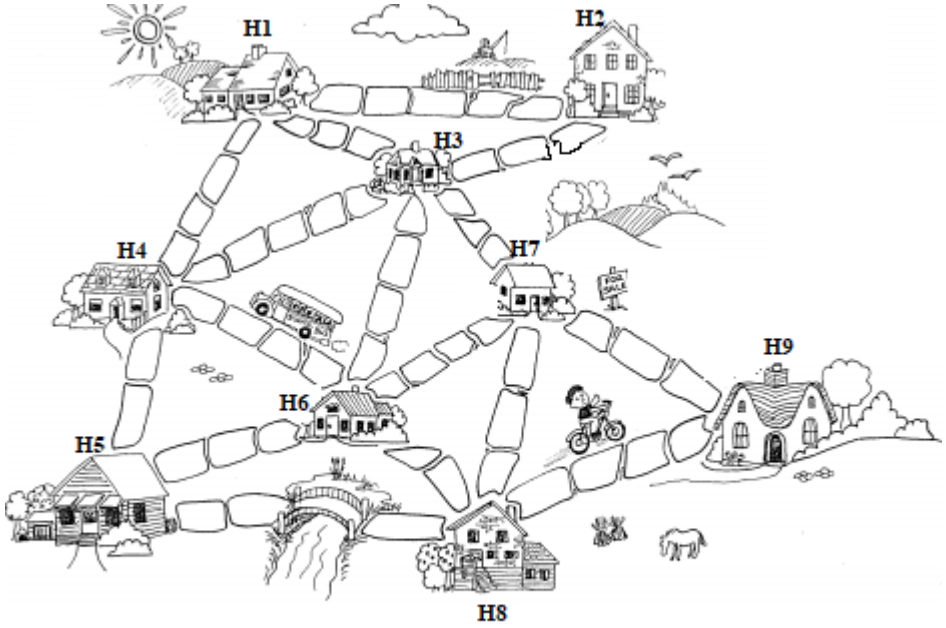
postfix String:

prefix String:

```
declare a stack of characters
while ( there are more characters in the infix string )
{
        read a character
        push the character on the stack
}/* end while */
declare rev string
while ( the stack is not empty )
{
        pop a character off the stack
        and append in rev string
}/* end while */
declare a stack opstk /*stack which stores operators*/
declare a postfix string
while (not end of rev string)
{
        symb = next input character;
        if (symb is an operand)
                add symb to the postfix string
        else {
                while (!empty(opstk) && precedenceChecker(stacktop(opstk),symb) )
                {
                        topsymb = pop(opstk);
                        add topsymb to the postfix string;
                 } /* end while */
                push(opstk, symb);
              } /* end else */
    } /* end while */
/* output any remaining operators */
while (!empty(opstk) )
{
        topsymb = pop(opstk);
        add topsymb to the postfix string;
} /* end while */
while ( there are more characters in the postfix string )
{
        read a character
        push the character on the stack
}/* end while */
declare prefix string
while ( the stack is not empty )
{
        pop a character off the stack
        and append in prefix string
}/* end while */
```

Question 6 [Marks 10]

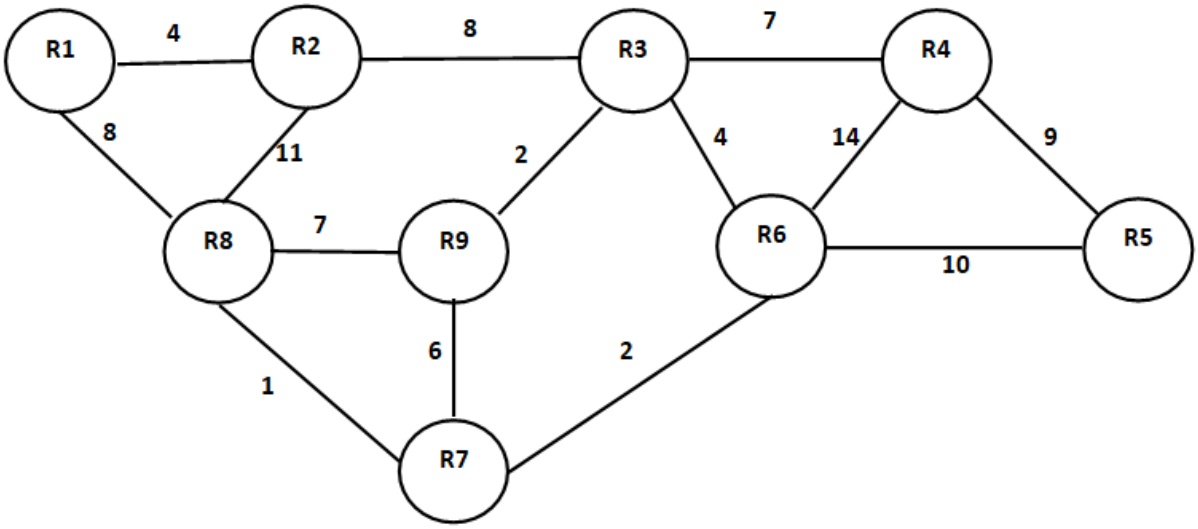
Once upon a time there was a city that had no roads. Getting around the city was particularly difficult after rainstorms because the ground became very muddy; cars got stuck in the mud and people got their shoes dirty. The mayor of the city decided that some of the streets must be paved, but didn't want to spend more money due to financial constraints. The mayor therefore specified two conditions:
1. Enough streets must be paved so that it is possible for everyone to travel from their house to anyone else's house only along paved roads, and
2. The paving should cost as little as possible.
Following is the layout of the city. The number of paving stones between houses represents the cost of paving that route. Applying **Kruskal algorithm**, find the best route that connects all the houses, but uses as few paving stones as possible. Justify why your offered solution is the optimal one. Redraw the resulting layout of the city showing the houses as well as the paving cost among them.



Note - Each block corresponds to a unit weight e.g. 3 blocks considered as weight 3. Similarly a bridge carries weight of 2 in the given scenario

Question 7 [Marks 10 + 5]

Given the following graph depicting an inter-city communication network comprising of 9 different routers interconnected through routes/links of variable bandwidth. In the graph, the weights shown next to the edges represent bandwidth of those communication links in Mbps. Suppose a routing protocol makes routing decisions on basis of bandwidth of communication links/paths **and higher the bandwidth of a link/path is, lower the cost it carries** [Note: use common sense in estimating the link cost]



(a) Write the pseudocode of a function to determine the best path from a given specified router (source) to all other routers in the network. Note the pseudocode must fulfill the bandwidth criteria mentioned earlier.

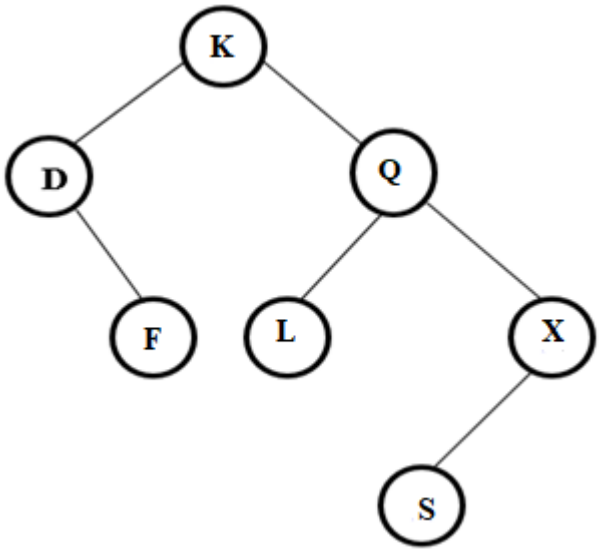
(b) Use Dijkstra's algorithm to determine the best route as per the bandwidth criteria from router **"R1"** to all other routers. Show each step of the algorithm in tabular form. Here's the table after the initial step: (∞ represents unavailability of a direct path)

Done	D[R2]	D[R3]	D[R4]	D[R5]	D[R6]	D[R7]	D[R8]	D[R9]
R1	4,R1	∞	∞	∞	∞	∞	8,R1	∞

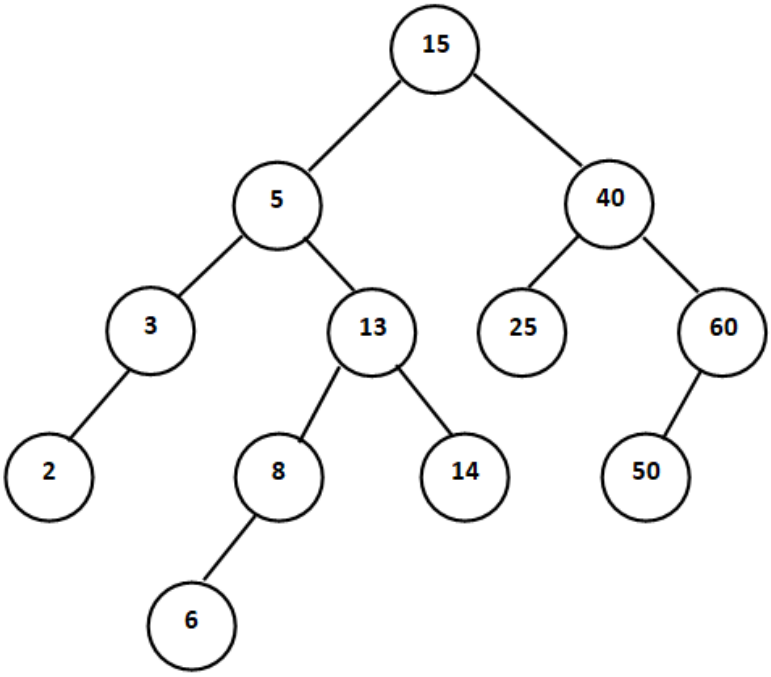
Note that along with the distance, the "previous" vertex is also shown. You may add additional rows in table when required.

Question 8 [Marks 8 + 2]

1. Attempt the following questions using concepts of AVL trees.
   a. Insert characters "**E**", "**R**", "**T**", "**U**", "**Z**" into the following AVL tree one at a time and draw the resulting AVL tree after each insertion. Also specify any sort of transformation that might be required to ensure the resulting tree is an AVL. [5 Marks]



   b. Given the following AVL tree, delete "**40**" and reconstruct the resulting AVL tree. Describe the step-by-step procedure in performing this deletion. [3 Marks]



2. Write C++ code of a recursive function to print values of all leaf nodes of a Binary tree in reverse order (i-e, all leaf nodes on right to be printed, followed by those on left side of tree). [2 Marks]