****

**BAHRIA UNIVERSITY (KARACHI CAMPUS**)

FINAL EXAMINATION –FALL SEMESTER – 2020

**Data structure & Algorithm: CSC-221**

Class: **BS(CS) – 3 A/B**  **(Morning)**

Course Instructor: **Lubna Siddiqui** Time Allowed: **2.5 Hours**

Date: **2nd Feb2021 Session: I**  Max Marks: **50**

**Student Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Enrolment #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Part A**

***Solve Part A on Question paper***

**Question no.1 (5 marks)**

Answer each of the following questions by choosing the best answer from the list below. Write alphabet as answer.

1. **Stack** A data structure used to convert an infix expression to postfix notation
2. A **Tree** collection of nodes with each one linked to its parent, and a single root on top
3. **O (log n)** The average case search time in a Binary Search Tree (BST)
4. **Circular Link list** A linked list in which the last reference points to the first reference.
5. **AVL** is used when the developer wants to control the tree height outside -1 to 1 range.
6. **Minimum spanning tree** of a weighted G is a spanning tree such that the sum of edge weights is minimized
7. **Quick Sort** algorithm is based on divide n conquer paradigm.
8. **Directed graphs** are often used to represent order-dependent tasks.
9. **Traversing**Accessing each record exactly once so that certain items in the record may be processed.
10. **Strictly binary tree** if every non-leaf node in a binary tree has non-empty left and right sub-trees.

|  |  |  |
| --- | --- | --- |
| a.           ArrayList | i.           Circular Link list | q.         O(n) |
| b.           AVL | j.         O(log n) | r.          Priority Queue |
| c.          Stack | k.          Quick Sort | s.          Double Linked List |
| d.          Queue | m.         Minimum spanning tree | t.          Strictly binary tree |
| e.           BFS | n.          LinkedList | u.      Directed graph |
| f . Cyclic graph | o. Tree | v. Insertion sort |
| g. Searching | p. Traversing | w. BST |

**Question no.2 (7 marks)**

Answer the following questions. Answer must be to the point

1. What happens in a stack if top= -1?

I f stack is -1 then it means stack is empty and is underflow condition

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Write the condition of overflow if queue is circular.

If( ( (front=rear+1) )| | ( (front==0) &&(rear=n-1) ) )

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. What queue is represented, if rear=6 and front=2 with N=10?

1. Quick sort and merge sort belong to divide and conquer technique. Give divide and conquer steps of both algorithms.

* **Quicksort** is a divide-and-conquer sorting algorithm in which division is dynamically carried out.
* Divide: Rearrange the elements and split the array into two sub arrays and an element in between such that so that each element in the left subarray is less than or equal the middle element and each element in the right subarray is greater than the middle element.
* Conquer: Recursively sort the two subarrays
* **Merge Sort** algorithm:
  + Divide the list into two roughly equal halves.
  + Sort the left half.
  + Sort the right half.
  + Merge the two sorted halves into one sorted list.

1. Can we do binary search on link list? Justify your answer

**Yes**, Binary search is possible on the linked list if the list is ordered and you know the count of elements in list.

Binary Search is usually fast and efficient for arrays because accessing the middle index between two given indices is easy and fast(Time Complexity O(1)). But memory allocation for the singly linked list is dynamic and non-contiguous, which makes finding the middle element difficult and time complexity becomes O(n).

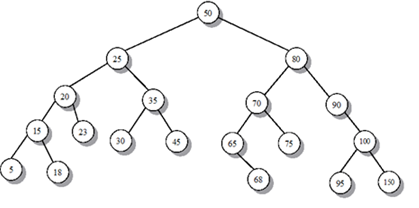
**Question no.3 (5 marks)**

|  |  |
| --- | --- |
| a. | Following tree is binary search tree or not. Justify your answer.    **Answer**  It is not a binary search tree because 58 is greater than 57 and should be inserted to the left of 59 |
| b.  c. | Is following binary tree complete or not? Justify your answer  **Yes it is a complete binary tree because all levels are filled.**  Is following binary Search Tree is balanced or not? Justify your answer.    **Answer**  It is a balanced tree because the balance factor of all nodes is between -1 to 1 range |
| d. | Give in order traversal of the above expression tree    **Answer**  In order(Left , Node, Right)  2 \* 3 / 2 – 1 + 5 \* 4 - 1 |
| e. | If there are 8191 nodes in complete binary tree. Find out the depth of complete binary tree  **Answer**  d=log(n+1)-1  d=log(8191+1)-1 |

**Part B**

**Question no.4 (6+4 marks)**

1. Consider following BST .

****

1. Give Pre order, in order and post order traversal of above tree.
2. Insert 54 in above BST
3. Delete 45 from above BST
4. Delete 80 from above BST
5. Insert the following keys in an empty AVL tree.

100,150,200,250,170,80,160,165,90

**Question no.5 (5+3 marks)**

a. Apply Breadth first search algorithm to following graph, also draw BFS tree.

b. Perform topological sort on following graph

**Question no.4 (2+2+2 marks)**

Consider a hash table of size 7 with hash function h(k) = k mod 7. Draw the table that results after inserting, in given order, the following values: 19, 26, 13, 48, 17 for each of the three scenarios below:

1. When collisions are handled by separate chaining?
2. When collisions are handled by linear probing?
3. When collisions are handled by quadratic probing?

**Question no.5 (3+3+3 marks)**

1. A pile of 100 books placed on a table. Sara arranged books from table to shelf in one-by-one fashion. Write an algorithm that prints number of books left on table.
2. Write a recursive algorithm to find if a given string is palindrome or not. Palindrome are string that are same forward and backward. For example, madam, rotor.

**void RString(char\* s, int len)**

**{**

**if (len <= 0)**

**{**

**cout << "loop" << endl;**

**return;**

**}**

**RString(s + 1, len - 1);**

**cout << \*s;**

**cout << endl;**

**}**

1. Write the code for following diagram



**Struct node\*{**

**Int data**

**Node\*next**

**}**

**Node\* head=NULL**

**insert Position(int n,int info)**

**{**

**Node\*ptr=head**

**Node\*temp=new node;**

**Temp->data=info**

**For(int i=1;i<n-1;i++){**

**Ptr=ptr->next;**

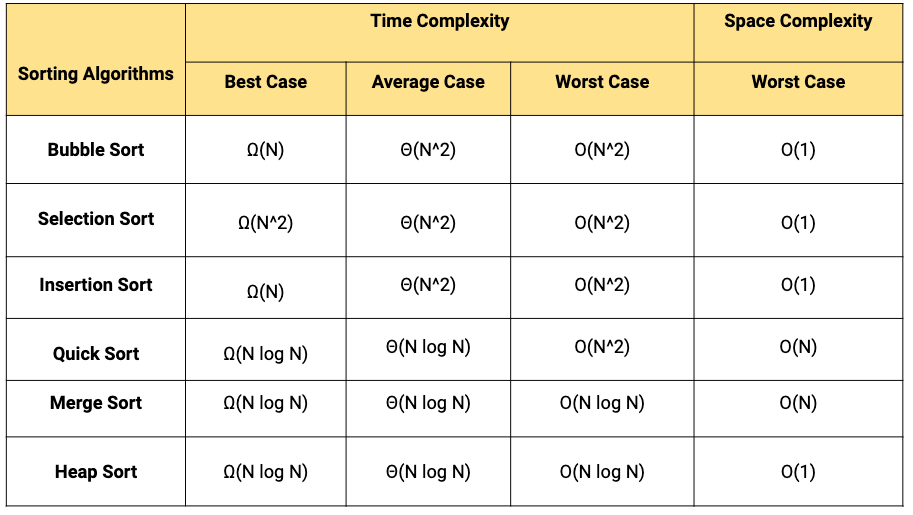
**}**

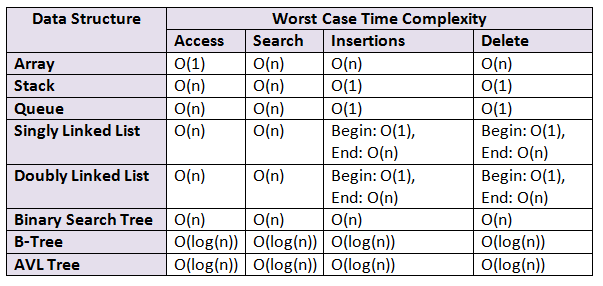
**Temp->next=ptr->next;**

**Ptr->next=temp;**

**}**

**NOTES:**





[**Queue data structure**](https://www.geeksforgeeks.org/queue-data-structure/) maintain two pointers, *front* and *rear*. The *front* points the first item of queue and *rear* points to last item.  
**enqueue()** This operation adds a new node after *rear*and moves *rear* to the next node.

**Insertion at Tail**  
**dequeue()** This operation removes the front node and moves *front* to the next node.

**Deletion from Head**

**Both operation at O(1)**

**Stack Operations:**

1. [**push()**](https://www.geeksforgeeks.org/stack-push-and-pop-in-c-stl/)**:** Insert a new element into stack i.e just inserting a new element at the beginning of the linked list.
2. [**pop()**](https://www.geeksforgeeks.org/stack-push-and-pop-in-c-stl/)**:** Return top element of the Stack i.e simply deleting the first element from the linked list.
3. **Both operation at O(1)**