Military Institute of Science and Technology
B.Sc. in Computer Science and Engineering
Quiz, Spring-2022
Subject: CSE-204, Data Structures and Algorithms-I Sessional

Time: 40 Minutes

Full Marks: 40

INSTRUCTIONS

a. Answer all the questions.
b. Each question contains 1 mark.
c. Fill up only one circle for the best possible answer for each question
d. Filling up wrong answer will deduct 0.5 marks

1 Identify the statement that is correct about static array

A. Size can't determined at compile time
B. Size can be modified at runtime
C. Size can be determined at run time
[✓ D.] Size can be determined at compile time

2

```
int x = Array[len-1];
for (int i = len-2; i >= 0; i--){
        Array[i+1] = Array[i];
}
Array[0] = x;
```

Identify the operation that has been performed in the code snippet mentioned above.

A. Rotate Left
B. Shift Left
[✓ C.] Rotate Right
D. Shift Right

3

```
static_array arr;
arr.insert_element_last(10);
arr.insert_element_first(20);
arr.insert_element_last(30);
arr.insert_element_last(40);
arr.insert_element_first(50);
arr.delete_element_last();
arr.delete_element_first();
arr.print();
```

Consider the code snippet mentioned above where the description of the functionalities of a static array list are described in the following table.

Function	**Description**
void insert_element_last(int x);	Inserts ***x*** at the end of the static array list
void insert_element_first(int x);	Inserts ***x*** at the beginning of the static array list
void delete_element_last( );	Deletes the element located at the end of the static array list
void delete_element_first( );	Deletes the element located at the beginning of the static array list
void print( );	prints all the elements of the static array list from beginning to end sequentially

Now if the size of the static array list is 10 then identify the output of the code snippet mentioned above.

[✓ A.] 20 10 30
B. 50 20 10 30
C. 50 20 10 30 40
D. 10 20 30

4 Identify the return type of malloc() or calloc() function.

[✓ A. void *] B. Pointer of allocated memory type

C. void ** D. int *

5 The ______ library function allocates a specified number of bytes and initializes them to zero. Identify the appropriate option for filling up the gap.

[✓ A. calloc( )] B. malloc( )

C. zeroloc( ) D. realloc( )

6 After execution of the following program, identify the memory segment where the variables will be stored.

```
int main() {
    int x;
    int *y = (int *) malloc(sizeof(int));
}
```

A. ***x*** and ****y*** both will be stored in stack segment [✓ B. ***x*** will be stored in stack segment and ****y*** on heap segment.]

C. ***x*** will be stored in data segment and ****y*** on heap segment. D. ***x*** and ****y*** both will be stored on heap segment

7 Identify the number of elements in a doubly linked list if ***head*** equals to ***tail***.

A. 1 B. 2

C. 3 [✓ D. Either 0 or 1]

8 Identify the time complexity of finding the maximum element of a singly linked list if the number of elements in the singly linked list is denoted by ***n***.

A. O(1) [✓ B. O(n)]

C. O($\log_2 n$) D. O(n^2)

9 Identify the operation of a singly linked list that is faster than the corresponding operation of an array. Note that the singly linked list maintains no ***tail*** pointer.

A. Inserting after last element B. Finding the middle element

[✓ C. Deleting the first element] [✓ D. Deleting the first element]

10 Identify the operation of a doubly linked list that executes in constant time. Note that the doubly linked list maintains both ***head***(leftmost) and a ***tail***(rightmost) pointer.

A. Finding the middle element B. Sorting the list

C. Rotating all the elements in the list to one position left [✓ D. Deleting the middle element]

11 Identify the data structure where applying Binary Search is not possible.

A. Array [✓ B. Linked list]

C. Vector D. B and C

12 Consider that the following operations are performed sequentially on a Stack. Identify the sequence in which the values have been removed from the stack:

- *push* (7);
- *push* (23);
- *pop*( );
- *push* (12);
- *push* (4);
- *pop*( );
- *pop*( );
- *push* (9);
- *pop*( );
- *pop*( );

A. 7 23 12 4 9 [✓ B. 23 4 12 9 7]

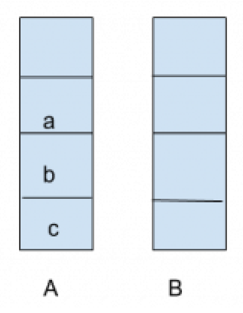
C. 23 7 4 12 9 D. 9 4 12 23 7

13 Consider a Stack whose maximum possible size is denoted by ***n***. Identify the runtime complexity to pop an element that was inserted in the Stack at the very beginning.

A. $O(1)$ B. $O(n^2)$

[✓ C. $O(n)$] D. $O(nlogn)$

14 Stack ***A*** has the entries as following sequence a, b, c (with 'a' on top), stack ***B*** is empty, as shown in the diagram below. An entry popped out of stack ***A*** can be printed or pushed to stack ***B***. An entry popped out of stack ***B*** can only be printed. In this arrangement, identify the sequence of a, b, c that is not possible to print.



A. b, a, c B. b, c, a

C. a, b, c [✓ D. c, a, b]

15 A circular queue has been implemented using a singly linked list and the following operations have been performed on the circular queue:
enqueue(33)
enqueue(44)
enqueue(11)
dequeue
enqueue(22)
enqueue(55)
dequeue
dequeue
Identify the final sequence of elements of the given circular queue.

A. 11 22 55 B. 55 22 11

[✓ C. 22 55] D. 33 44

16 Identify the data structure required for performing Breadth First Traversal on a graph.

A. Tree B. Array

C. Stack [✓ D.] Queue

17 A queue data structure has been implemented with an array. Identify the problem that may occur.

A. Overflow B. Underflow

[✓ C.] Both D. None

18 Identify the principle that a queue follows.

[✓ A.] First In First Out B. Last In First Out

C. First In Last Out D. Last In Last Out

19 Identify the traversal that is equivalent toDepth First Search performed on a Binary Tree.

[✓ A.] Pre-order Traversal B. Post-order Traversal

C. Level-order Traversal D. In-order Traversal

20 Identify the time complexity of Breadth First Search performed on a complete graph with $\boldsymbol{n}$ number of vertices.

A. $O(\boldsymbol{n})$ [✓ B.] $O(\boldsymbol{n}^2)$

C. $O(\boldsymbol{n}^3)$ D. $O(\boldsymbol{n}^2\log_2\boldsymbol{n})$

21 Identify the fastest method to visit all the nodes of a Binary Search Tree starting from the root.

A. Preorder traversal B. Postorder traversal

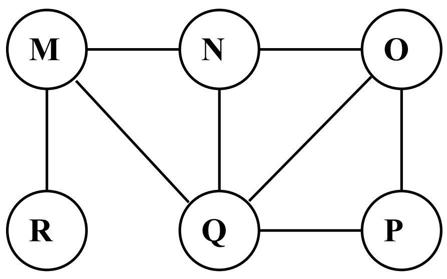
C. Inorder traversal [✓ D.] All of them

22 Identify the result that will be generated after performing Depth First Search traversal on a connected undirected graph.

A. Binary Tree [✓ B.] Tree

C. Graph with back edges D. Forest

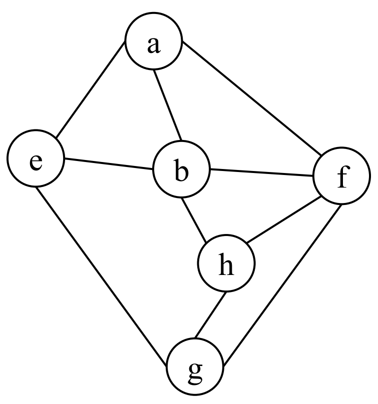
23 The BFS Algorithm has been applied on the following graph. Identify the valid possible order of visiting the vertices.



A. MNOPQR B. NQMPOR

C. QMNROP [✓ D.] POQNMR

24 Consider the following graph



Among the following sequences:
(I) a b e g h f
(II) a b f e h g
(III) a b f h g e
(IV) a f g h b e

Identify the valid depth first traversals of the above graph

A. I, II, IV
B. I, IV
C. II, III, IV
[✓ D. I, III, IV]

Consider the following definition of **Node** class for a Binary Search Tree ***B***.

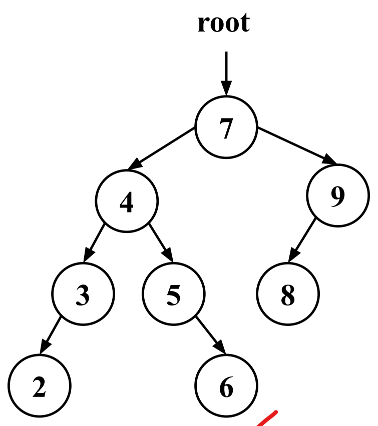
```
class Node{
public:
    int key;
    Node *left, *right, *parent;
}*root;
```

The ***left*** pointer points to the left child, the ***right*** pointer points to the right child and the ***parent*** pointer points to the parent of a **Node** in ***B***. ***root*** pointer points to the initial node of ***B***. If a **Node** does not have any left child then the ***left pointer*** points to NULL and if the **Node** does not have any right child then the ***right*** pointer points to NULL. The ***parent*** pointer of ***root*** **Node** points to NULL. Now based on the above scenario, answer the questions from 25 to27 .

25 Consider the following function ***f*** that operates on ***B***.

```
int f(int k, Node *cur = root)
{
    if(cur==NULL)   return -1;
    if(k==0)   return cur->key;
    return   max( f(k-1, cur->left), f(k-1, cur->right) );
}
```

Identify the value of *f*(2) for the following figure of ***B*.**



A. 9
[✓ B. 8]
C. 6
D. -1

26 Consider the following function ***print***

```
void print(Node *cur){
   f(cur->right);
   printf(“%d”, cur->key);
   f(cur->left);
}
```

If the ***root*** of ***B*** is passed as ***cur*** to the *print* function then identify the task of the function.

A. Printing the inorder sequence of the BST

B. Printing all the keys in descending order of the BST

C. Printing all the keys in ascending order of the BST

[✓ D.] Leading to runtime error

27 Consider the following function. Identify the incident that will occur if *xyz*(***root***) is called from the main function when ***B*** has only one **Node** [***root***].

```
void xyz(Node *node){
     Node *par = node->parent;
     if(node->key < par->key){
          par->left = NULL;
     }
     else{
          par->right = NULL;
     }
     free(node);
}
```

A. The node will be inserted in the tree

B. The node will be deleted from the tree

C. Parent of the node will be the new root of the tree

[✓ D.] It will generate a runtime error.

28 Consider a Binary Search Tree with ten nodes and the keys of the tree are marked from 1 to 10. If no two nodes have the same key then identify the inorder successor of 5 in the Binary Search Tree.

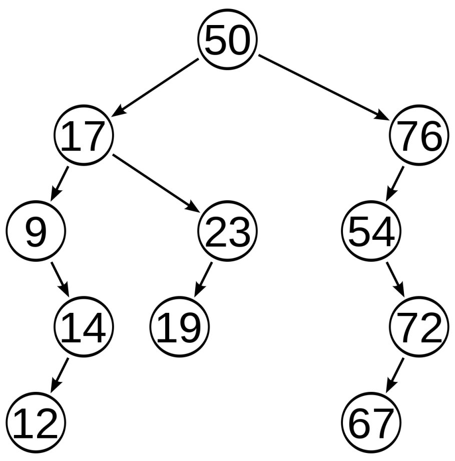
A. 1

[✓ B.] 6

C. 4

D. Can not be identified from the given information

29 Identify the parent of 17 after deleting the root node from the BST.



A. 23

B. 76

[✓ C.] 54

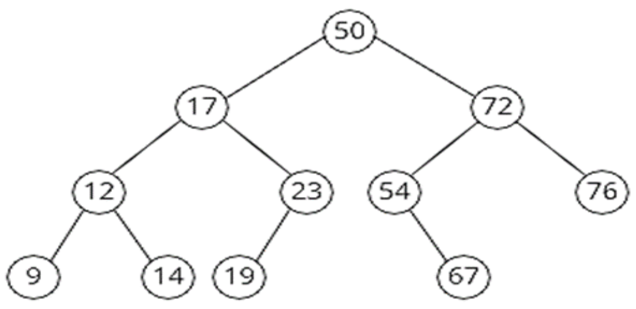
D. 67

30 Identify the number of possible Binary Search Trees with three nodes and the keys of the tree are marked from 0 to 2. If no two nodes have the same key then identify the total number of possible Binary Search Trees.

A. 3 B. 4

[✓ C.] 5 D. 6

31 Considering the following Binary Search Tree. Now if we insert 46 in this tree, then identify the node that will be the parent of 46.



A. 54 **B.** 76

[✓ C.] 23 D. 17

32 Identify the correct order for the Merge Sort.

A.
- i. Solve left Sub Problem
- ii. Merge
- iii. Solve right Sub Problem

B.
- i. Solve left Sub Problem
- ii. Solve right Sub Problem
- iii. Merge

C.
- i. Merge
- ii. Solve left Sub Problem
- iii. Solve right Sub Problem

D.
- i. Merge
- ii. Solve right Sub Problem
- iii. Solve left Sub Problem

33 Consider the left sub array After solving: 2 4 7 9
Consider the right sub array After solving: 1 6 8 12
If the above two solved subproblems are to be merged then identify the element that will be in index 4 in the new merged list.

A. 8 B. 7

C. 6 D. 4

34 Suppose in a divide & conquer algorithm, the array of length ***n*** is divided into **three equal** parts and solved recursively. Then we combine these by iterating ***n*** times. Identify the recurrence formula.

A. $T(n) = 2T(n/2) + n$ B. $T(n) = 3T(n/3) + n$

C. $T(n) = 2T(n/3) + n$ D. $T(n) = 3T(n/2) + n$

35 Consider the following array ***arr*** that is sorted in ascending order.

arr	10	12	16	21	40	48	60	65
	0	1	2	3	4	5	6	7

The function *find*(***arr***, ***x***) returns the index of ***x*** in ***arr*** using binary search algorithm. Identify the value of ***x*** for which the *find*(***arr***, ***x***) requires the highest number of iterations.

A. 10 B. 21

C. 60 [✓ D.] 65

36 Identify the value by which 4^{35} is calculated in the Binary-exponentiation method.

A. 4^{34} B. 4^{16}

C. 2^{34} D. None of them

37 Consider the following function *f_sqrt* written in C++ language.

```
int f_sqrt(int x, int k=4){
    int low = 0, high = x, mid;
    for(int i=1; i<=k; i++){
        mid = (low + high)/2;
        if(x < mid*mid) high = mid;
        else    low = mid;
    }
    return mid;
}
```

Identify the value of *f_sqrt*(25).

A. 0 B. 3

[✓ C.] 4 D. 5

38 Consider the following function:

```
int find(int arr[], int x, int n, int low, int high);
```

This function returns the index of the last occurrence of ***x*** in the array ***arr*** sorted in ascending order using binary search. ***n*** denotes the number of elements in ***arr*** and the array is indexed from 0 to ***n***-1. The function works recursively. At each recursive step ***mid*** is calculated as (***low*** + ***high***)/2. Now identify the statement that will be executed `if(arr[mid] == x && arr[mid+1] == x)`

[✓ A.] return find(arr, x, n, mid+1, high); B. return find(arr, x, n, low, mid+1);

C. return mid; D. return -1;

39 Consider the following function *find* written in C language.

```
int find(int arr[], int x, int low, int high){
    if(low > high)   return -1;
    int mid = (low + high)/2;
    if(x == arr[mid])  return mid;
    if(x < arr[mid])   return find(arr, x, low, mid-1);
    else   return find(arr, x, mid+1, high);
    return mid;
}
```

arr	10	12	16	21	40	48	60	65	88	96
	0	1	2	3	4	5	6	7	8	9

For finding 48 from ***arr*** the initial call is `find(arr, 48, 0, 9);` Identify the next recursive call.

A. `find(arr, 48, 0, 4);` B. `find(arr, 48, 0, 3);`

C. `find(arr, 48, 4, 9);` [✓ D.] `find(arr, 48, 5, 9);`

40 Identify the appropriate sequence of inserting keys in a Binary Search Tree for which the height of the tree becomes balanced.

A. 10 20 30 40 50 60 70 80 B. 45 32 31 30 18 10 7 2

C. 5 5 5 5 5 5 [✓ D.] 4 2 6 1 3 7 8