## PART A: OBJECTIVE QUESTIONS (20 Marks)

*Part A consists of 20 objective questions. Each question carries 1 mark. Choose the correct answer and write your answer in the answer booklet.*

1. In a Binary Search algorithm applied on a sorted list, what is the initial step?
   A. Compare the target with the last element in the list.
   B. Compare the target with the first element in the list.
   C. Compare the target with the element at the middle index of the list.
   D. Split the list into two equal parts without any comparison.

2. What is the time complexity for a sequential search in an unsorted dataset and sorted dataset when running in the worst-case scenario?
   A. O(log n) and O(log n)
   B. O(n) and O(n)
   C. O(n) and O(log n)
   D. O(log n) and O(n log n)???

3. A sorted array B={15, 22, 27, 36, 42, 50, 65, 72} contains a key = 36. Using the Sequential Search method, how many iterations (p-value) are required before the key is located?
   A. 2
   B. 3
   C. 4
   D. 5

4. Given an array C={10, 18, 25, 33, 41, 56, 63, 71, 80}, with key = 63, determine the middle elements (corresponding array values) checked during the first two iterations of a Binary Search?
   A. 33 and 56
   B. 56 and 63
   C. 41 and 63
   D. 56 and 80

[See next page]

5. Which type of queue technique experiences the right-drift problem?
   A. Circular Queue
   B. Simple Queue
   C. Linear linked-list Queue
   D. Circular linked-list Queue

6. Which of the following is **NOT** true about queue implementation using an array?
   A. The size of the queue is fixed once declared?
   B. Enqueue and dequeue operations have a time complexity of O(1)
   C. Always suffer from the right-drift problem
   D. Provide the simplest queue implementation technique

7. In a circular queue implementation using an array and a counter flag, what happens when the front has the value of 0 and the back has value of array's maximum index value?
   A. The queue is empty
   B. The queue is full
   C. The queue contains only one element
   D. The queue is in an invalid state

8. What type of queue implementation applies the following instructions to enqueue a new item?

```
newItem->next = backPtr->next;
backPtr->next = newItem;
```

   A. Linear Linked-list Queue
   B. Doubly Linked-list Queue
   C. Circular Linked-list Queue
   D. Doubly Circular Linked-list Queue

9. Which of the following is **NOT** an example of a Stack in a real-world application?
   A. Web browser history
   B. Undo operation in a text editor
   C. Recursive function call in a program
   D. Car wash waiting turn simulation

[See next page]

10. Given the following sequence of operations on an empty stack:

    `push(9), push(19), pop(), push(90), pop(), pop()`

    What is the value of the last element popped?

    A. 9

    B. 19

    C. 90

    D. Stack is empty

11. Given the infix expression `p * (3 + b) / 7`. What is the equivalent prefix expression?

    `A. /p +*3b7`

    `B. * p / + 3b7`

    `C. * p +/ 3b7`

    D. `/ * p + 3b7`

12. Which of the following is **TRUE** about implementing a stack using a linked list?

    A. The top of the stack corresponds to the first node/item added in the singly linked list.

    B. The stack can grow or shrink dynamically without a size limit.

    C. It requires isFull() method before push operation.

    D. The time complexity O(1) for push and O(n) for pop?

13. Which of the following is **TRUE** about linked list?

    A. Search operation is faster when using linked list
    B. Linked list operation is simple compare to array
    C. The number of items is limited to the size of the linked list
    D. Virtually has unlimited number of items can be stored by using linked list

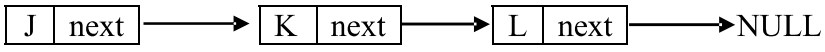
14. Given the nodes T, A, and Z are connected as follows: T connected to A, A connected to Z, and Z connected to T. What is the type of linked list applied to those nodes?

    A. Singly linked list
    B. Doubly linked list
    C. Circular linked list
    D. Doubly circular linked list

[See next page]

15. Given the nodes J and L are connected to each other as below:


Choose the correct instructions to connect node K to form new linked structure as follows:




A. `J->next = K; L->next = NULL;`

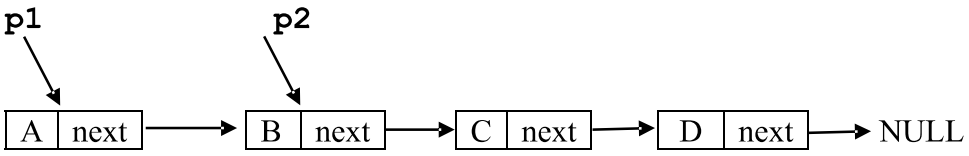
B. `J->next = K; K->next = L;`

C. `K->next = L; L->next = NULL;`

D. `K->next = L; L->next = J;`

16. Given a class definition and linked list structure as follows:

```
class Node {
  public:
    char item;
    Node* next;
}
```




What is the output of the below C++ instructions?

```
cout << p1->next->data;
cout << p1->next->next->data;
cout << p2->data;
cout << p2->next->next->data;
```

A. B C B D
B. B C B C
C. A B A C
D. C D B C

[See next page]

17. In a binary tree, which traversal produces a sorted sequence of values?

    A. Preorder

    B. Postorder

    C. Inorder

    D. Level-order

18. What is the height of a tree with only a single node?

    A. 0

    B. 1

    C. 2

    D. Undefined

19. A full binary tree is a tree in which:

    A. Every node has 0 or 2 children

    B. All levels except possibly the last are completely filled

    C. All nodes are arranged from left to right

    D. The height difference between left and right subtrees is $\leq 1$

20. What type of traversal is implemented by the following code?

```
void traverse(Node* node) {
  if (node == NULL) return;
  traverse(node->left);
  traverse(node->right);
  cout << node->data << " ";
}
```

    A. Preorder

    B. Postorder

    C. Inorder

    D. No specific order is applied

[See next page]

## PART B: STRUCTURED QUESTIONS (80 Marks)

*Part B consists of 5 structured questions. Answer all questions in the answer booklet.*

**Question 1 (15 MARKS)**

a) Explain the difference between internal search and external search in terms of where the data is stored and processed, and the reasons for choosing one method over the other.

**(2 marks)**

b) An array **ODDS** contains 15 consecutive odd numbers arranged in ascending order:

| **Index** | **[0]** | **[1]** | **[2]** | **[3]** | **[4]** | **[5]** | **[6]** | **[7]** | **[8]** | **[9]** | **[10]** | **[11]** | **[12]** | **[13]** | **[14]** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Array | 1 | 3 | 5 | 7 | 9 | 11 | 13 | 15 | 17 | 19 | 21 | 23 | 25 | 27 | 29 |

Using an **improved** sequential search strategy, identify the **number of comparisons, final index**, and **array value** for the following search keys:

**(6 marks)**

| **Search Key** | **Num. of Comparisons** | **Final Index** | **Array Value** |
|---|---|---|---|
| 15 | | | |
| 6 | | | |

c) Using the **Binary Search** algorithm, determine whether the number 25 exists in the **ODDS** array in question 1 (b) above. Show the search process by filling in the table below:

**(5 marks)**

| **Iteration** | **LEFT** | **RIGHT** | **MIDPOINT** | **OODS[MIDPOINT]** | **FOUND** |
|---|---|---|---|---|---|
| 1 | | | | | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | | | | | |

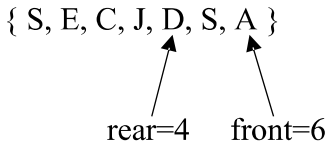
[See next page]

d) Analyze and compare the efficiency of the search techniques used in Question 1 (b) and Question 1 (c). Explain how the dataset size affects their performance.

**(2 marks)**

**Question 2 (15 MARKS)**

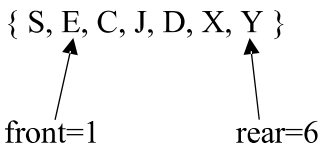
a) Given the current state of **circular queue array** as below:




i) What is the total number of items currently in the queue?

**(1 mark)**

ii) Write a series of `dequeue()` and `enqueue(item)` operations that will change the state of the **circular queue array** as follows:




**(4 marks)**

b) Describe the role of the **`frontPtr`** and **`backPtr`** pointers in a linked list implementation of a queue.

**(2 marks)**

c) The code segment below shows the implementation of a **`Queue`** class using a linked list. Complete the **`enqueue`** method implementations (lines 13-19).

**(4 marks)**

```
01  class Node {
02    public:
03      int data;
04      Node(int num) { data = num; }
05  };
06
07
```

```
08  class Queue {
09    public:
10      Node *frontPtr = NULL, *backPtr = NULL;
11
12      void enqueue (int element) {
13        Node *newNode = new Node(element);
14        if (backPtr == NULL) { // queue is empty
15          ____________________;
16          ____________________;
17        } else { // queue is not empty
18          ____________________;
19          ____________________;
20        }
21      }
22
23      ~Queue() {
24        frontPtr = NULL;
25        backPtr = NULL;
26      }
27  };
```

d) The code segment above (lines 23-26) shows an improper implementation of the `Queue` class destructor. Fix the code with the correct implementation.

**(4 marks)**

**Question 3 (15 MARKS)**

a) Given an array-based stack with a maximum size of **3** with the following operations to be performed:

```
1. push(15)
2. push(8)
3. x = pop()
4. push(5)
5. y = pop() + pop()
6. push(x)
7. push(y)
8. push(pop() + pop())
9. z = pop() + pop()
```

i) Complete the diagram below to show the changes in the stack content for each operation performed.

**(6 marks)**

[See next page]

| Operations: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| Stack Content | | | | | | | | | |
| | | 8 | | | | | | | |
| | 15 | 15 | 15 | | | | | | |

ii) Is there an error in the sequence of operations? If so, identify the operation that causes the error and explain whether it is an overflow or underflow error.

**(2 marks)**

b) Given the table with two rows of expressions below, determine their type (postfix/prefix) and convert them to their original infix form.

**(3 marks)**

| Expression | Type | Infix |
|---|---|---|
| A B C / – D * E + | | |
| + * - 5 2 / 4 3 | | |

c) Given the following postfix expression:

**9 1 5 * + 7 / 4 –**

Evaluate the postfix expression using stack operations. Write your answer and show the evaluation by completing the table below.

**(4 marks)**

| Postfix | Ch | Op | Oprn1 | Oprn2 | Result | Stack |
|---|---|---|---|---|---|---|
| **9 1 5 * + 7 / 4 -** | | | | | | |
| **1 5 * + 7 / 4 -** | **9** | | | | | **# 9** |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

**Question 4 (20 MARKS)**

a) Given Program-4.1 to implement linked list as follows:

```
1  // Program-4.1
2
3  class Node {
4    public:
5      int value;
6      Node* next;
7
8      Node(int v) {
9        value = v;
10     }
11 };
12
13 int main() {
14
15     // declare head and tail as pointer type variables of Num class
16     Node *head=NULL, *tail=NULL;
17
18     // create linked list [6 marks]
19     for (int i = 10; i <= 30; i += 10) {
20       Node *n = _____________;
21
22       if (head == _______) {
23         ____________________;
24
25       } else {
26         ____________________;
27       }
28
29       tail = ___________;
30       ___________________________________;
31     }
32
33     // add node 15 between nodes 10 & 20 [2 marks]
34     Node *n15 = new Node(15);
35     ____________________________;
36     ____________________________;
37
38     // delete node 30 and replace it with node 25 [3 marks]
39     Node *n25 = new Node(25);
40     Node *n30 = ________;
41     Node *n20 = ____________________;
42     n20->next = ________;
43     n25->next = ________;
44     tail = ________;
45     delete ________;
46
47     // print values and delete all nodes [5 marks]
48     Node *n = _________;
49     Node *prev = NULL;
```

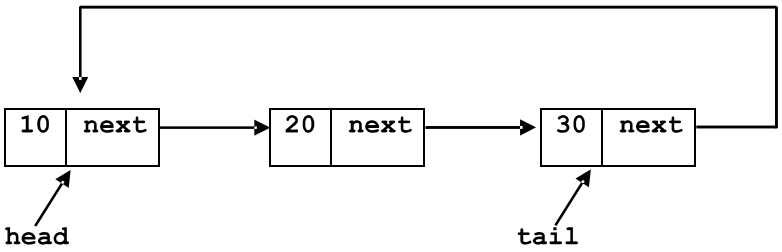
[See next page]

```
50
51      while (n != __________) {
52          _____________________;
53          _____________________;
54          _____________________;
55          _____________________;
56      }
57
58      // print and delete last node/tail
59      cout << n->value << " ";
60      delete n;
61  }
```

Based on the Program-4.1 above do the followings:

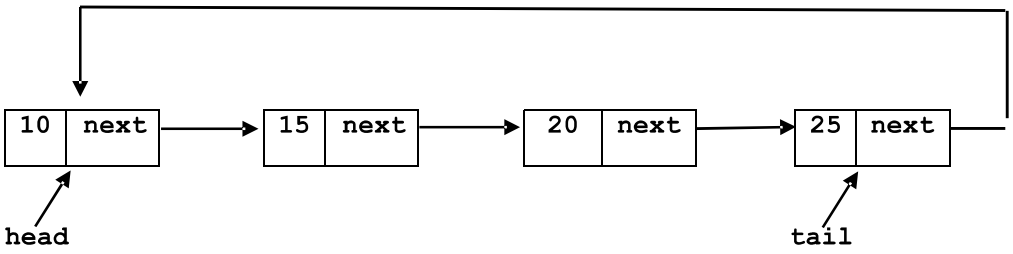
i) Fill in the required code (lines 20-30) to create the linked list structure shown below:

**[6 marks]**




ii) Assume that the linked list structure in (i) has been successful constructed, fill in the blank lines (35-45) to change the linked list structure as below:

**[5 marks]**




iii) Fill in the blank lines (48-56) to print the values and delete all nodes from the final linked list structure shown in (ii) above.

**[5 marks]**

b) Given Program-4.2 to create linked list as follows:

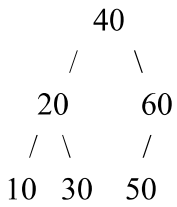
```
 1 // Program-4.2
 2
 3 class Alpha {
 4   public:
 5     char item;
 6     Alpha *n, *p;
 7
 8     Alpha(char c) {
 9         item = c;
10     }
11 };
12
13 int main() {
14
15     Alpha *a1 = new Alpha('A');
16     Alpha *a2 = new Alpha('D');
17     Alpha *a3 = new Alpha('S');
18
19     a2->n = a3;
20     a3->n = a1;
21     a1->n = NULL;
22
23     a1->p = a3;
24     a3->p = a2;
25     a2->p = NULL;
26 }
```

Draw the structure of the linked list constructed by Program-4.2 above, and specify its name.

**[4 marks]**

**Question 5 (15 MARKS)**

Consider a binary tree with the following structure:




a) Write the outputs of preorder, inorder, and postorder traversals of the given tree.

**(6 marks)**

b) Calculate the height of the tree and explain how the height of a binary tree is determined

**(2 marks)**

[See next page]

c) Insert new nodes with the value 32 and 55 into the tree, maintaining the Binary Search Tree property, draw the updated tree structure.

**(3 marks)**

d) Draw the new tree structure after removing nodes 20 and 60 from the tree.

**(3 marks)**

e) Check the final tree structure either it is full, complete and balanced.

**(1 mark)**

End of question