Application Book

Data Structure CS203

2nd Level

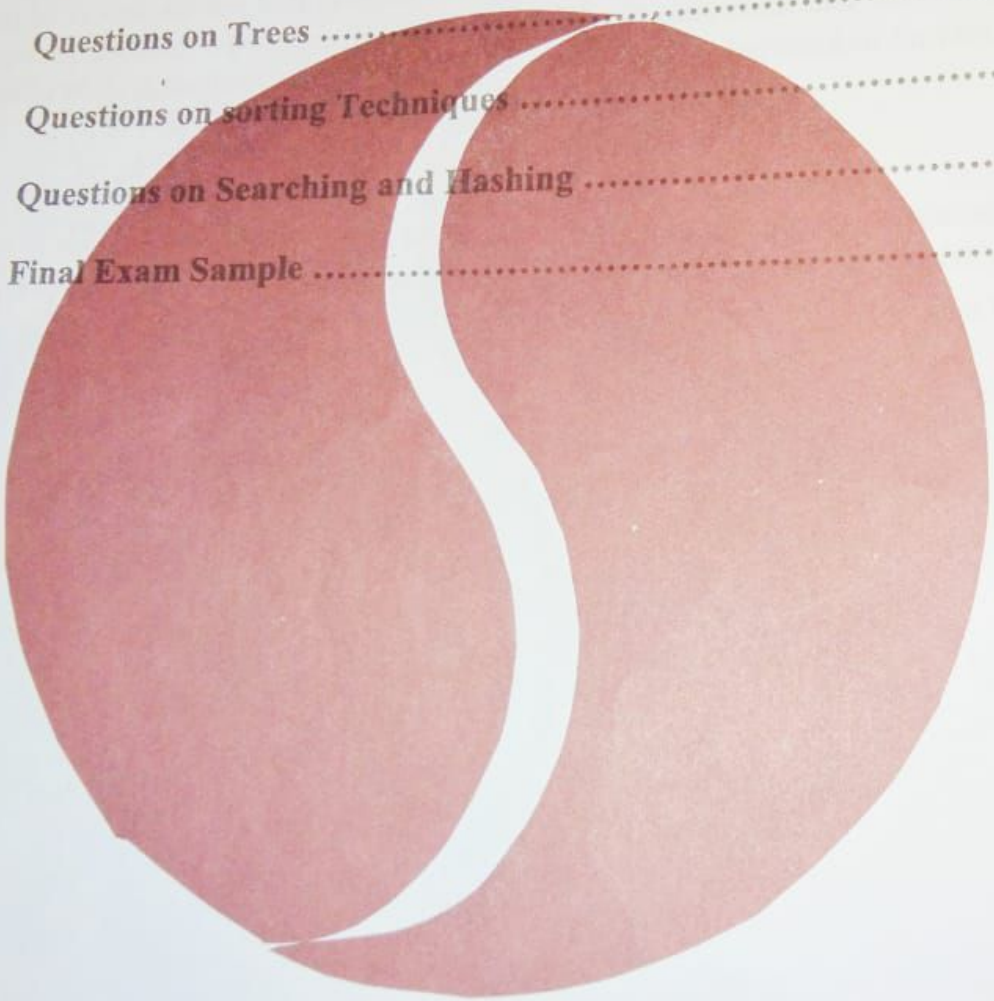
2019-2020

Introduction

This book is prepared as a complementary part for the lecture notes of the course "Introduction to computers". The main objective of this book is to train the student to go through the different applications of the course. The book contains examples of the questions and exercises and the model answer for some of them. The student is required to refer to the references of the course and the lecture notes to find out the answer of the unsolved questions and exercises. These exercises aim to assess the intended learning outcomes relevant to knowledge and understanding as well as the intellectual skills of this course.

The book contains also some examples for the practical applications that should be practiced in the Lab sessions. Some of these applications are introduced and their model answers are given as a guide for the students.

Table of Contents



Questions on Pointers

1- Write a program to print the address of a variable whose value is input from user.

```
#include<iostream>
using namespace std;

int main()
{
    int x;
    cout << "Enter a number\n";
    cin >> x;
    cout << "Address is "<<&x<<"\n";
    return 0;
}
```

2- Write a program to print the value of the address of the pointer to a variable whose value is input from user.

```
#include<iostream>
using namespace std;

int main()
{
    int x, *y;
    cout << "Enter a number\n";
    cin >> x;
    y = &x;
    cout << "Value of the address of pointer of "<< x << " is "<< &y<<"\n";
    return 0;
}
```

3- Write a program to print a number which is entered from keyboard using pointer.

```
#include<iostream>
using namespace std;

int main()
{
    int x;
    cout << "Enter a number\n";
    cin >> x;
    cout << *(&x) << "\n";
    return 0;
}
```

4- **Write a function which will take pointer and display the number on screen. Take number from user and print it on screen using that function.**

```
#include<iostream>
using namespace std;

void print(int *a)
{
    cout << *a << "\n";
}

int main()
{
    int x;
    cout << "Enter a number\n";
    cin >> x;
    print(&x);
    return 0;
}
```

5- **What is the output from the following code?**

```
int main()
{
    int num[5];
    int* p;
    p = num;
    *p = 10;
    p++;
    *p = 20;
    p = &num[2];
    *p = 30;
    p = num + 3;
    *p = 40;
    p = num;
    *(p + 4) = 50;
    for (int i = 0; i < 5; i++)
        cout << num[i] << ", ";
    return 0;
}
```

Answer:

10, 20, 30, 40, 50,

What is the output from the following code?

```
int main()
{
    int i = 20;
    int* ptr = &i;
    (*ptr)++;
    int j = 15;
    ptr = &j;
    cout << i;
    return 0;
}
```

Answer:
21

What is the output from the following code?

```
int main()
{
    int arr[] = { 4, 5, 6, 7 };
    int* p = (arr + 1);
    cout << *arr + 10;
    return 0;
}
```

Answer:
14

What is the output from the following code?

```
int main()
{
    int a[] = { 10, 20, 30 };
    cout << *a + 1;
}
```

Questions on Linked List

What are the main differences between Arrays and Linked Lists

	Array	Linked List
Basic	It is a consistent set of a fixed number of data items.	It is an ordered set comprising a variable number of data items.
Size	Specified during declaration.	No need to specify; grow and shrink during execution.
Storage Allocation	Element location is allocated during compile time.	Element position is assigned during run time.
Order of the elements	Stored consecutively	Stored randomly
Accessing the element	Direct or randomly accessed, i.e., Specify the array index or subscript.	Sequentially accessed, i.e., Traverse starting from the first node in the list by the pointer.
Insertion and deletion of element	Slow relatively as shifting is required.	Easier, fast and efficient.
Memory required	less	More
Memory Utilization	Ineffective	Efficient

1- Which of the following points is/are true about Linked List data structure when it is compared with array

a) Arrays have better cache locality that can make them better in terms of performance
b) It is easy to insert and delete elements in Linked List
c) Random access is not allowed in a typical implementation of Linked Lists
d) The size of array has to be pre-decided, linked lists can change their size any time
e) All of the above

2- The following C function takes a single-linked list of integers as a parameter and rearranges the elements of the list. The function is called with the list containing the integers 1, 2, 3, 4, 5, 6, 7 in the given order. What will be the contents of the list after the function completes execution?

```
struct node
{
  int value;
  struct node *next;
};
void rearrange(struct node *list)
{
  struct node *p, * q;
  int temp;
  if ((!list) || !list->next)
      return;
  p = list;
  q = list->next;
  while(q)
  {
     temp = p->value;
    p->value = q->value;
    q->value = temp;
    p = q->next;
    q = p?p->next:0;
  }
}
```

Answer

2,1,4,3,6,5,7

3- In the worst case, What is the number of comparisons needed to search a singly linked list of length n for a given elements?

Answer

n

4. Consider the following function to traverse a linked list.

```
void traverse(struct Node *head)
{
   while (head->next != NULL)
   {
       cout<< head->data);
       head = head->next;
   }
}
```

Which of the following is **FALSE** about above function?

a. The function may crash when the linked list is empty
b. The function doesn't print the last node when the linked list is not empty
c. The function is implemented incorrectly because it changes head

5. Consider you have the class file single_llist.h file

```
#include<iostream>
#include<cstdio>
#include<cstdlib>
using namespace std;
/*
 * Node Declaration
 */
struct node
{
    int info;
    struct node *next;
}*start;

/*
 * Class Declaration
 */
class single_llist
{
    public:
        node* create_node(int);
        void insert_begin();
        void insert_pos();
        void insert_last();
        void delete_pos();
        void search();
        void display();
        single_llist() {
            start = NULL;
        }
};
```

Implement the following functions

1. Create node
2. Insert Node at beginning
3. Insert node at last

4. Insert node at position
5. Search Element
6. Display Linked List

Create Node

```
node *single_llist::create_node(int value)
{
    struct node *temp, *s;
    temp = new(struct node);
    if (temp == NULL)
    {
        cout<<"Memory not allocated "<<endl;
        return 0;
    }
    else
    {
        temp->info = value;
        temp->next = NULL;
        return temp;
    }
}
```

Insert at beginning:

```
void single_llist::insert_begin()
{
    int value;
    cout<<"Enter the value to be inserted: ";
    cin>>value;
    struct node *temp, *p;
    temp = create_node(value);
    if (start == NULL)
    {
        start = temp;
        start->next = NULL;
    }
    else
    {
        p = start;
        start = temp;
        start->next = p;
```

```
    }
    cout<<"Element Inserted at beginning"<<endl;
}
```

Insert node at end

```
void single_llist::insert_last()
{
    int value;
    cout<<"Enter the value to be inserted: ";
    cin>>value;
    struct node *temp, *s;
    temp = create_node(value);
    s = start;
    while (s->next != NULL)
    {
        s = s->next;
    }
    temp->next = NULL;
    s->next = temp;
    cout<<"Element Inserted at last"<<endl;
}
```

Search a linked list

```
void single_llist::search()
{
    int value, pos = 0;
    bool flag = false;
    if (start == NULL)
    {
        cout<<"List is empty"<<endl;
        return;
    }
    cout<<"Enter the value to be searched: ";
    cin>>value;
    struct node *s;
    s = start;
    while (s != NULL)
    {
        pos++;
        if (s->info == value)
```

```
            {
                flag = true;
                cout<<"Element "<<value<<" is found at position
 "<<pos<<endl;
            }
            s = s->next;
        }
        if (!flag)
            cout<<"Element "<<value<<" not found in the list"<<endl;
}
```

Display the list

```
void single_llist::display()
{
    struct node *temp;
    if (start == NULL)
    {
        cout<<"The List is Empty"<<endl;
        return;
    }
    temp = start;
    cout<<"Elements of list are: "<<endl;
    while (temp != NULL)
    {
        cout<<temp->info<<"->";
        temp = temp->next;
    }
    cout<<"NULL"<<endl;
}
```

Stacks and Queues

Choose the correct answer

1- The postfix form of the expression (A+ B)*(C*D- E)*F / G is?
   a) AB+ CD*E - FG /**
   b) AB + CD* E - F **G /
   c) AB + CD* E - *F *G /
   d) AB + CDE * - * F *G /

2- The data structure required to check whether an expression contains balanced parenthesis is?
   a) Stack
   b) Queue
   c) Array
   d) Tree

3- The postfix form of A*B+C/D is?
   a) *AB/CD+
   b) AB*CD/+
   c) A*BC+/D
   d) ABCD+/*

4- Which data structure is needed to convert infix notation to postfix notation?
   a) Branch
   b) Tree
   c) Queue
   d) Stack

5- The prefix form of A-B/ (C * D ∧ E) is?
   a) -/*∧ACBDE
   b) -ABCD*∧DE
   c) -A/B*C∧DE
   d) -A/BC*∧DE

6- What is the result of the following operation
Top (Push (S, X))
   a) X
   b) Null
   c) S
   d) None

7- The prefix form of an infix expression p + q - r * t is?

a) + pq - *rt

b) - +pqr * t

c) - +pq * rt

d) - + * pqrt

8- Which data structure is used for implementing recursion?

a) Queue

b) Stack

c) Array

d) List

9- The result of evaluating the postfix expression 5, 4, 6, +, *, 4, 9, 3, /, +, * is?

a) 600

b) 350

c) 650

d) 588

10- Convert the following infix expressions into its equivalent postfix expressions

(A + B ∧ D)/(E - F)+G

a) (A B D ∧ + E F - / G +)

b) (A B D + ∧ E F - / G +)

c) (A B D ∧ + E F/- G +)

d) None

11- Convert the following Infix expression to Postfix form using a stack

x + y * z + (p * q + r) * s, Follow usual precedence rule and assume that the expression is legal.

a) xyz*+pq*r+s*+

b) xyz*+pq*r+s+*

c) xyz+*pq*r+s*+

d) none

12- Consider the linked list implementation of a stack. Which of the following node considered as Top of the stack?

a) First node
b) Last node
c) Any node
d) Middle node

13- Consider the following operation performed on a stack of size 5.

```
Push(1);
Pop();
Push(2);
Push(3);
Pop();
Push(4);
Pop();
Pop();
Push(5);
```

After the completion of all operation, the no of element present on stack are

a) 1
b) 2
c) 3
d) 4

14- The type of expression in which operator succeeds its operands is?

a) Infix Expression
b) pre fix Expression
c) postfix Expression
d) None

15- If the elements "A", "B", "C" and "D" are placed in a stack and are deleted one at a time, in what order will they be removed?

a) ABCD
b) DCBA
c) DCAB
d) ABDC

16- Elements can be retrieved by index in .....?

A. linked lists
B. linear arrays
C. both of above
D. none of above

17- Efficiency of an algorithm is measured by

A. Time and Capacity complexity
B. Time and Space complexity
C. Speed and Space complexity
D. Speed and Capacity complexity

18- State True or False about array and linked list

I. Retrieval of element will be faster in array than link list?
II. Search operation in both array and link list are leaner.

A. True, False
B. False, True
C. False, False
D. True, True

19- In which data structure memory is contiguous

A. Array
B. Link list
C. Both
D. None

20- Consider the following pseudocode that uses a stack

```
declare a stack of characters
while ( there are more characters in the word to read )
{
   read a character
   push the character on the stack
}
while ( the stack is not empty )
{
   pop a character off the stack
   write the character to the screen
}
```

What is output for input "geeksquiz"?

1. qeeksquizgeeksquiz
2. ziuqskeeg
3. geeksquiz
4. ziuqskeegziuqskeeg

21- The result evaluating the postfix expression 10 5 + 60 6 / * 8 – is

284
213
142
71

22- Which of the following permutation can be obtained in the same order using a stack assuming that input is the sequence 5, 6, 7, 8, 9 in that order?

7, 8, 9, 5, 6
5, 9, 6, 7, 8
7, 8, 9, 6, 5
9, 8, 7, 5, 6

23- The following C function takes a single-linked list of integers as a parameter and rearranges the elements of the list. The function is called with the list containing the integers 1, 2, 3, 4, 5, 6, 7 in the given order. What will be the contents of the list after the function completes execution?

```
struct node
{
    int value;
    struct node *next;
};
void rearrange(struct node *list)
{
   struct node *p, * q;
   int temp;
   if ((!list) || !list->next)
     return;
  p = list;
  q = list->next;
  while(q)
  {
      temp = p->value;
      p->value = q->value;
```

```
            q->value = temp;
            p = q->next;
            q = p?p->next:0;
        }
    }
```

Answer:

2, 1, 4, 3, 6, 5, 7

24- What is stack and queue in C++?

Stack and Queue both are the non-primitive data structures. The main differences between **stack and queue** are that stack uses LIFO (last in first out) method to access and add data elements whereas **Queue** uses FIFO (First in first out) method to access and add data elements

25- Following is C like pseudo code of a function that takes a Queue as an argument, and uses a stack S to do processing.

```
void fun(Queue *Q)
{
    Stack S;  // Say it creates an empty stack S

    // Run while Q is not empty
    while (!isEmpty(Q))
    {
        // deQueue an item from Q and push the dequeued item to S
        push(&S, deQueue(Q));
    }

    // Run while Stack S is not empty
    while (!isEmpty(&S))
    {
        // Pop an item from S and enqueue the poppped item to Q
        enQueue(Q, pop(&S));
    }
}
```

Answer:

Reverse the queue

26- List some applications of Queue Data Structure?

1- When a resource is shared among multiple consumers.
2- When data is transferred asynchronously (data not necessarily received at same rat between two processes.
3- Load balancing

27- Suppose a circular queue of capacity (n – 1) elements is implemented with an array c n elements. Assume that the insertion and deletion operation are carried out usin REAR and FRONT as array index variables, respectively. Initially, REAR = FRONT 0. The conditions to detect queue full and queue empty are

Suppose we start filling the queue.

Let the maxQueueSize (Capacity of the Queue) is 4.
So the size of the array which is used to implement
this circular queue is 5, which is n.

In the beginning when the queue is empty, FRONT and REAR
point to 0 index in the array.

REAR represents insertion at the REAR index.
FRONT represents deletion from the FRONT index.

```
enqueue("a"); REAR = (REAR+1)%5; ( FRONT = 0, REAR = 1)
enqueue("b"); REAR = (REAR+1)%5; ( FRONT = 0, REAR = 2)
enqueue("c"); REAR = (REAR+1)%5; ( FRONT = 0, REAR = 3)
enqueue("d"); REAR = (REAR+1)%5; ( FRONT = 0, REAR = 4)
```

Now the queue size is 4 which is equal to the maxQueueSize.
Hence overflow condition is reached.

Now, we can check for the conditions.

When Queue Full :

(REAR+1)%n = (4+1)%5 = 0

FRONT is also 0.

Hence (REAR + 1) %n is equal to FRONT.

When Queue Empty :

REAR was equal to FRONT when empty (because in the starting
before filling the queue FRONT = REAR = 0)

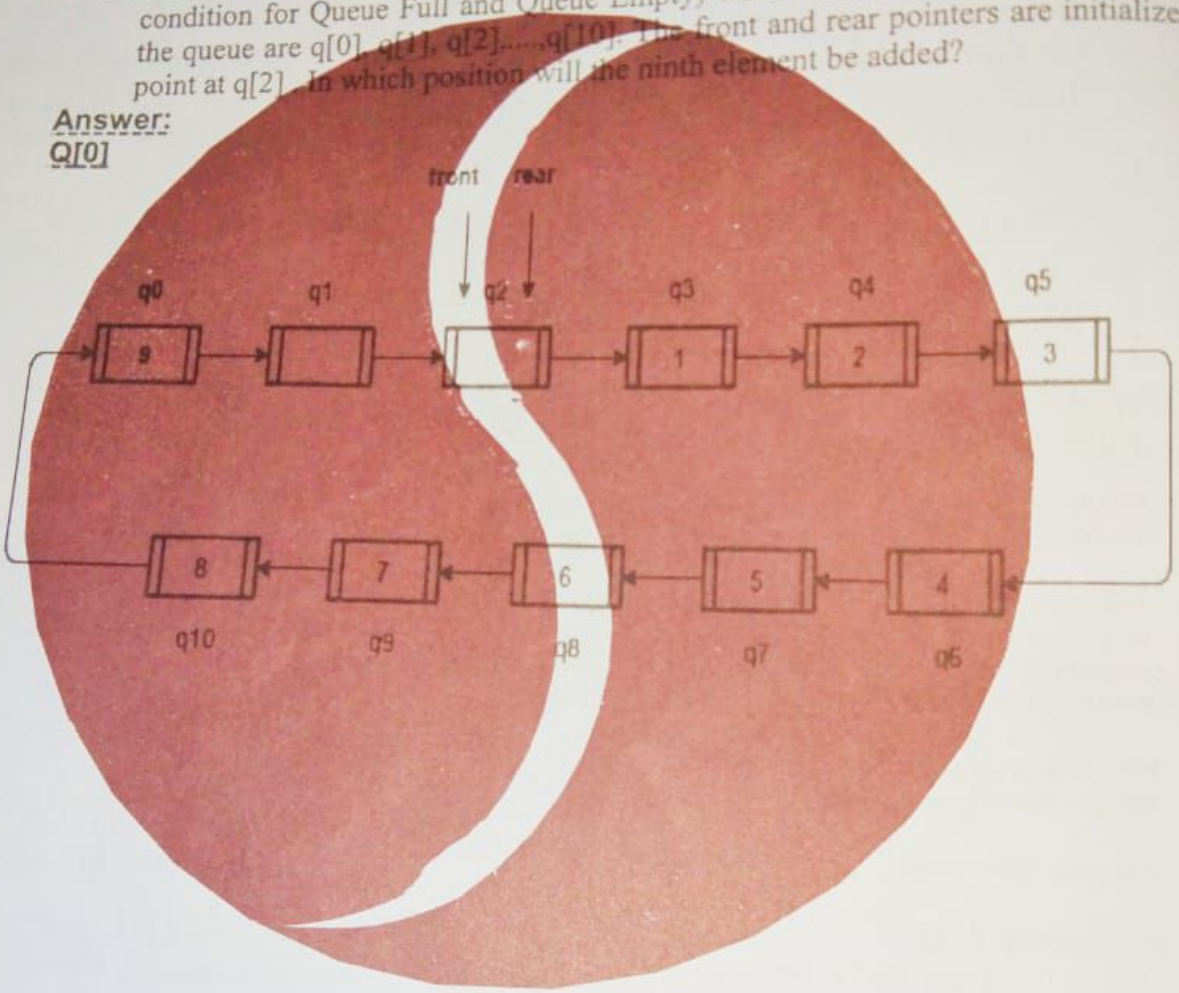
Answer

Full: (REAR+1) mod n == FRONT, empty: REAR == FRONT

29- Consider a standard Circular Queue 'q' implementation (which has the same condition for Queue Full and Queue Empty) whose size is 11 and the elements of the queue are q[0], q[1], q[2].....q[10]. The front and rear pointers are initialized to point at q[2]. In which position will the ninth element be added?

Answer:

Q[0]



Questions on Trees

Choose the correct Answer

1. The number of edges from the root to the node is called _____ of the tree.
   a) Height
   b) Depth
   c) Length
   d) Width
2. The number of edges from the node to the deepest leaf is called _____ of the tree.
   a) Height
   b) Depth
   c) Length
   d) Width
3. What is a full binary tree?
   a) Each node has exactly zero or two children
   b) Each node has exactly two children
   c) All the leaves are at the same level
   d) Each node has exactly one or two children
4. What is a complete binary tree?
   a) Each node has exactly zero or two children
   b) A binary tree, which is completely filled, with the possible exception of the bottom level, which is filled from right to left
   c) A binary tree, which is completely filled, with the possible exception of the bottom level, which is filled from left to right
   d) A tree In which all nodes have degree 2
5. Which of the following is not an advantage of trees?
   a) Hierarchical structure
   b) Faster search
   c) Router algorithms
   d) Undo/Redo operations in a notepad
6. In a full binary tree if number of internal nodes is I, then number of leaves L are?
   a) $L = 2*I$
   b) $L = I + 1$
   c) $L = I - 1$
   d) $L = 2*I - 1$

7. In a full binary tree if number of internal nodes is I, then number of nodes N are?
   a) N = 2*I
   b) N = I + 1
   c) N = I – 1
   d) N = 2*I + 1

8. Which of the following graph traversals closely imitates level order traversal of a binary tree?
   a) Depth First Search
   b) Breadth First Search
   c) Depth & Breadth First Search
   d) Binary Search

Both level order tree traversal and breadth first graph traversal follow the principle that visit your neighbors first and then move on to further nodes.

9- In a binary search tree, which of the following traversals would print the numbers in the ascending order?
   a) Level-order traversal
   b) Pre-order traversal
   c) Post-order traversal
   d) In-order traversal

In a binary search tree, a node's left child is always lesser than the node and right child is greater than the node, hence an in-order traversal would print them in a non decreasing fashion.

10- Which of the following is false about a binary search tree?
   a) The left child is always lesser than its parent
   b) The right child is always greater than its parent
   c) The left and right sub-trees should also be binary search trees
   d) In order sequence gives decreasing order of elements

In order sequence of binary search trees will always give ascending order of elements. Remaining all are true regarding binary search trees.

What is the speciality about the inorder traversal of a binary search tree?
a) It traverses in a non increasing order
b) It traverses in an increasing order
c) It traverses in a random fashion
d) It traverses based on priority of the node
As a binary search tree consists of elements lesser than the node to the left and the ones greater than the node to the right, an inorder traversal will give the elements in an increasing order.

What are the conditions for an optimal binary search tree and what is its advantage?
a) The tree should not be modified and you should know how often the keys are accessed, it improves the lookup cost
b) You should know the frequency of access of the keys, improves the lookup time
c) The tree can be modified and you should know the number of elements in the tree before hand, it improves the deletion time
d) The tree should be just modified and improves the lookup time
For an optimal binary search The tree should not be modified and we need to find how often keys are accessed. Optimal binary search improves the lookup cost.

11- Construct a binary search tree with the below information.
The preorder traversal of a binary search tree 10, 4, 3, 5, 11, 12.

Preorder Traversal is 10, 4, 3, 5, 11, 12. Inorder Traversal of Binary search tree is equal to ascending order of the nodes of the Tree. Inorder Traversal is 3, 4, 5, 10, 11, 12. The tree constructed using Preorder and Inorder traversal is

12- What will be the height of a balanced full binary tree with 8 leaves?

a) 8
b) 5
c) 6
d) 4

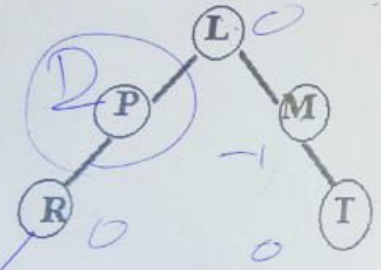
A balanced full binary tree with l leaves has height h, where h = log2l + 1.
So, the height of a balanced full binary tree with 8 leaves = log28 + 1 = 3 + 1 = 4.

13- The balance factor of a node in a binary tree is defined as _____

a) addition of heights of left and right subtrees
b) height of right subtree minus height of left subtree
c) height of left subtree minus height of right subtree
d) height of right subtree minus one

For a node in a binary tree, the difference between the heights of its left subtree and right subtree is known as balance factor of the node.

14- Figure below is a balanced binary tree. If a node inserted as child of the node R, how many nodes will become unbalanced?



a) 2
b) 1
c) 3
d) 0

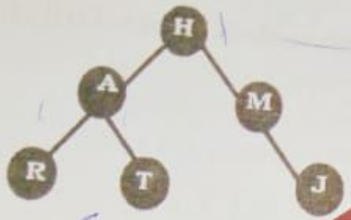
Only the node P will become unbalanced, with balance factor +2.

15- A binary tree is balanced if the difference between left and right subtree of every node is not more than ____

a) 1
b) 3
c) 2
d) 0

In a balanced binary tree the heights of two subtrees of every node never differ by more than 1.

16- Is This is a balanced binary tree? yes If every node in the tree is balanced, then it's a balanced tree.



17- Balanced binary tree with n items allows the lookup of an item in _____ worst-case time.

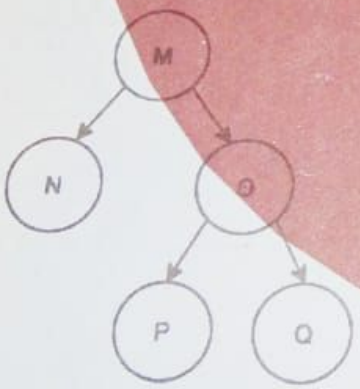
a) O(log n)
b) O(nlog 2)
c) O(n)
d) O(1)

Searching an item in balanced binary is fast and worst-case time complexity of the search is O(log n).

9. Construct a binary tree by using postorder and inorder sequences given below.

Inorder: N, M, P, O, Q

Postorder: N, P, Q, O, M

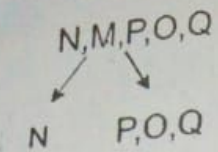


Postorder Traversal: N, P, Q, O, M

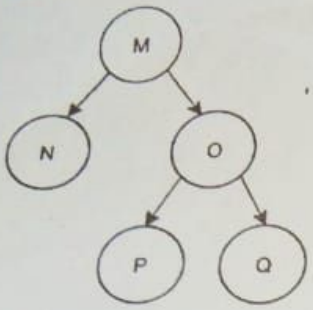
Inorder Traversal: N, M, P, O, Q

Root node of tree is the last visiting node in Postorder traversal. Thus, Root Node = 'M'.

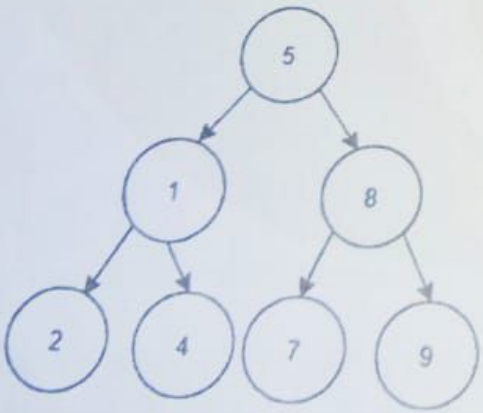
The partial tree constructed is:



The second last node in postorder traversal is O. Thus, node P becomes left child of node O and node Q becomes right child of node Q.

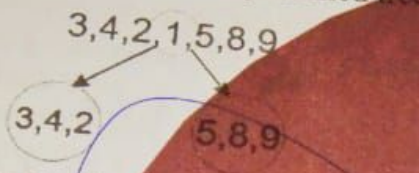


9- Construct a binary search tree by using postorder sequence given below.
Postorder: 2, 4, 3, 7, 9, 8, 5?
Explanation: Postorder sequence is 2, 4, 3, 7, 9, 8, 5.
Inorder sequence is the ascending order of nodes in Binary search tree. Thus, Inorder sequence is 2, 3, 4, 5, 7, 8, 9. The tree constructed using Postorder and Inorder sequence is

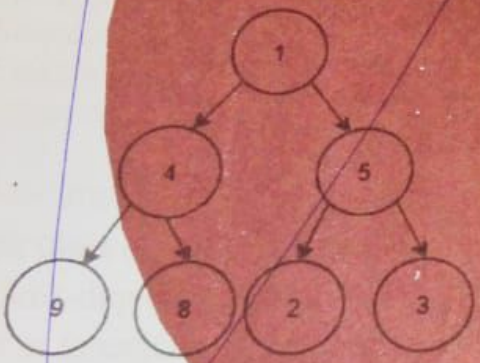


10- Construct a binary tree using inorder and level order traversal given below.
Inorder Traversal: 3, 4, 2, 1, 5, 8, 9
Level Order Traversal: 1, 4, 5, 9, 8, 2, 3

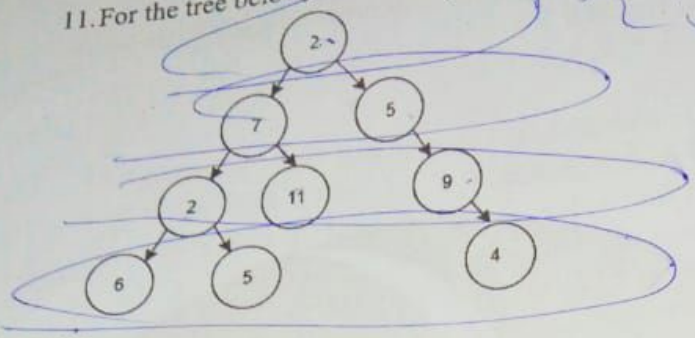
Explanation: Inorder Traversal: 3, 4, 2, 1, 5, 8, 9
Level Order Traversal: 1, 4, 5, 9, 8, 2, 3
In level order traversal first node is the root node of the binary tree.
Thus the partially formed tree is:



In level order traversal, the second node is 4. Then, node 3 becomes left child of node 4 and node 2 becomes right child of node 4. Third node of level order traversal is 8. Then, node 5 becomes left child of node 8 and node 9 becomes right child of node 8. Thus, the final tree is:



11. For the tree below, write the in-order traversal, the level-order traversal.



Answer

In-order traversal follows LNR(Left-Node-Right). 6, 2, 5, 7, 11, 2, 5, 9, 4

Level order traversal follows a breadth first search approach. 2, 7, 5, 2, 11, 9, 6, 5, 4

12. Which are the two standard ways of traversing a graph? Explain them with an example of each.

Answer

The two ways of traversing a graph are as follows:-

The depth-first traversal

of a graph is like the depth-first traversal of a tree. Since a graph has no root, when we do a depth-first traversal, we must specify the vertex at which to begin. Depth-first traversal of a graph visits a vertex and then recursively visits all the vertices adjacent to that node. The catch is that the graph may contain cycles, but the traversal must visit every vertex at most once. The solution to the problem is to keep track of the nodes that have been visited, so thatthe traversal does not suffer the fate of infinite recursion

The breadth-first traversal

of a graph is like the breadth-first traversal of a tree. Breadth-first tree traversal first visits all the nodes at depth zero (i.e., the root), then all the nodes at depth one, and so on. Since a graph has no root, when we do a breadth-first traversal, we must specify the vertex at which to start the traversal. Furthermore, we can define the depth of a given vertex to be the length of the shortest path from the starting vertex to the given vertex. Thus, breadth-first traversal first visits the starting vertex, then all the vertices adjacent to the starting vertex, and the all the vertices adjacent to those, and so on.

Questions on Sort techniques (Selection & insertion) Sort

1. Which of the following sorting algorithms should be preferred so that the number of swap operations are minimized in general?
   Heap Sort
   Insertion Sort
   Selection sort
   Merge Sort

2. Which sorting algorithm will take least time when all elements of input array are identical? Consider typical implementations of sorting algorithms.
   Heap Sort
   Insertion Sort
   Selection sort
   Merge Sort

3. Which one of the following in-place sorting algorithms needs the minimum number of swaps?
   Heap Sort
   Insertion Sort
   Selection sort
   Merge Sort

4. How many comparisons are needed to sort an array of length 5 if a straight selection sort is used and array is already in the opposite order?
   1
   5
   10
   20

5. **Write an algorithm for selection sort.**

1. Repeat step 2 and 3 varying j from 0 to n-2 //Repeat for n-1 passes
2. Find the index of the minimum value in arr[j] to arr[n-1]
   a. Set min_index=j
   b. Repeat step c varying i from j+1 to n-1
   c. If arr[i]<arr[min_index]: min_index = i

3. swap arr[j] with arr[min_index]

6. Write algorithm for Insertion Sort?

```
INSERTION-SORT(A)
  for i = 1 to n
       key ← A [i]
       j ← i - 1
        while j > = 0 and A[j] > key
                A[j+1] ← A[j]
                j ← j - 1
       End while
       A[j+1] ← key
End for
```

7. What are the correct intermediate steps of the following data set when it is being sorted with the Insertion sort?
15,20,10,18

15,20,10,18 -- 10,15,20,18 -- 10,15,18,20 -- 10,15,18,20

15,18,10,20 -- 10,18,15,20 -- 10,15,18,20 -- 10,15,18,20

15,10,20,18 -- 15,10,18,20 -- 10,15,18,20

10, 20,15,18 -- 10,15,20,18 -- 10,15,18,20

8. Consider the following lists of partially sorted numbers. The double bars represent the sort marker. How many comparisons and swaps are needed to sort the next number. [1 3 4 8 9 || 5 2]

2 comparisons, 3 swaps

3 comparisons, 2 swaps

4 comparisons, 3 swaps

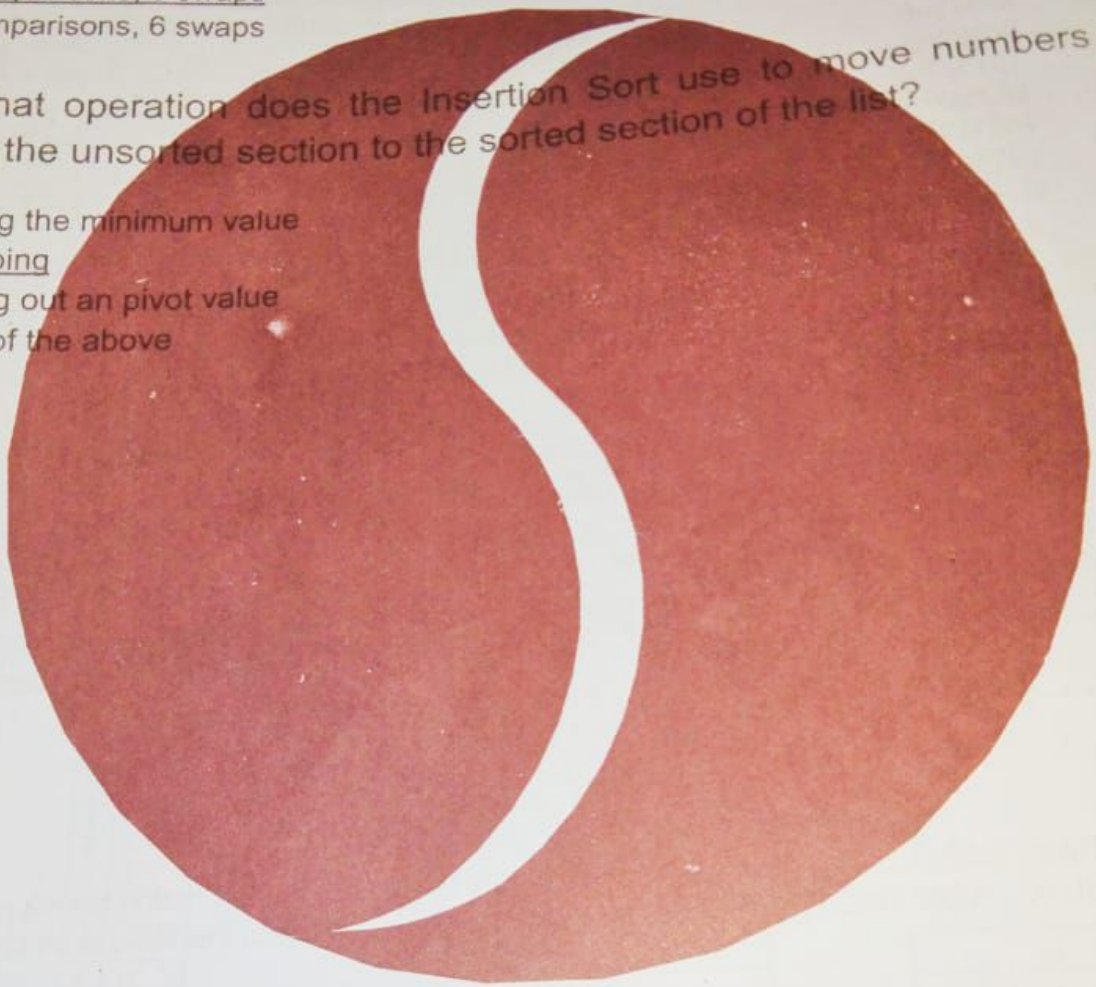
3 comparisons, 4 swaps

9. Consider the following lists of partially sorted numbers. The double bars represent the sort marker. How many comparisons and swaps are needed to sort the next number. [1 3 4 5 8 9 || 2]

5 comparisons, 4 swaps
4 comparisons, 5 swaps
6 comparisons, 5 swaps
5 comparisons, 6 swaps

10. What operation does the Insertion Sort use to move numbers from the unsorted section to the sorted section of the list?

Finding the minimum value
Swapping
Finding out an pivot value
None of the above



1. The following values are to be stored in a hash table25, 42, 96, 101, 102, 162, 197Describe how the values are hashed by using division method of hashing witha table size of 7. Use chaining as the method of collision resolution.

Answer:

Table size=7

25%7=4

42%7=0

96%7=5

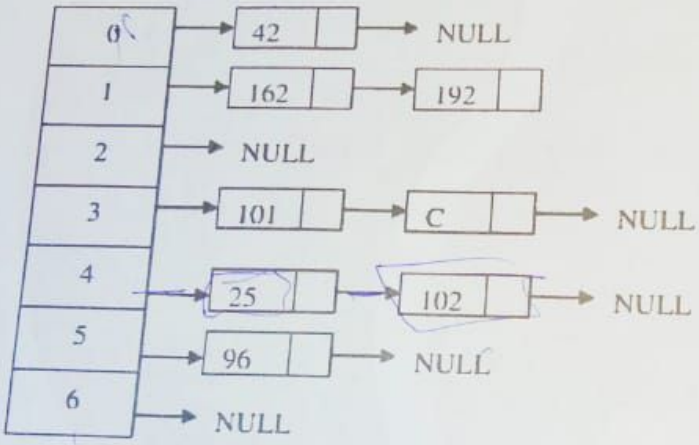
101%7=3

102%7=4

162%7=1

197%7=1

So collision resolution can be resolved as follows:



2. A hash table of length 10 uses open addressing with hash function h(k)=k mod 10, and linear probing. After inserting 6 values into an empty hash table, the table is as shown below.

0	
1	
2	42
3	23
4	34
5	52
6	46
7	33
8	
9	

Which one of the following choices gives a possible order in which the key values could have been inserted in the table?

a) 46, 42, 34, 52, 23, 33
b) 34, 42, 23, 52, 33, 46
c) 46, 34, 42, 23, 52, 33
d) 42, 46, 33, 23, 34, 52

How many different insertion sequences of the key values using the hash function **h(k) = k mod 10** and linear probing will result in the hash table shown above?

0	
1	
2	42
3	23
4	34
5	52
6	46
7	33
8	
9	

a) 10
b) 20
c) 30
d) 50

3. The keys 12, 18, 13, 2, 3, 23, 5 and 15 are inserted into an initially empty hash table of length 10 using open addressing with hash function h(k) = k mod 10 and linear probing. What is the resultant hash table?

	(A)	(B)	(C)	(D)
0				
1				
2	2	12	12	12, 2
3	23	13	13	13, 3, 23
4			2	
5	15	5	3	5, 15
6			23	
7			5	
8	18	18	18	18
9			15	

a) A
b) B
c) C
d) D

4. Consider a hash table of size seven, with starting index zero, and a hash function (3x + 4)mod7. Assuming the hash table is initially empty, which of the following is the contents of the table when the sequence 1, 3, 8, 10 is inserted into the table using closed hashing? Note that '_' denotes an empty location in the table.

   a) 8, _, _, _, _, _, 10
   b) 1, 8, 10, _, _, _, 3
   c) 1, _, _, _, _, _,3
   d) 1, 10, 8, _, _, _, 3

5. Consider a hash table of size seven, with starting index zero, and a hash function (3x + 4)mod7. Assuming the hash table is initially empty, which of the following is the contents of the table when the sequence 1, 3, 8, 10 is inserted into the table using closed hashing? Note that '_' denotes an empty location in the table.

a) 8, _, _, _, _, _, 10
b) 1, 8, 10, _, _, _, 3
c) 1, _, _, _, _, _,3
d) 1, 10, 8, _, _, _, 3

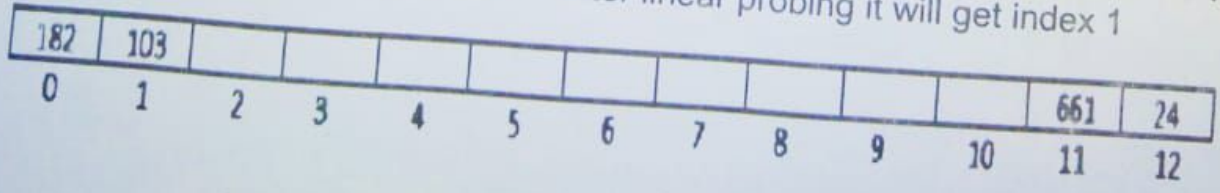
6. Given the following input (4322, 1334, 1471, 9679, 1989, 6171, 6173, 4199) and the hash function x mod 10, which of the following statements are true? i. 9679, 1989, 4199 hash to the same value ii. 1471, 6171 hash to the same value iii. All elements hash to the same value iv. Each element hashes to a different value

   a) i only
   b) ii only
   c) i and ii only
   d) iii or iv

7. Consider a 13 element hash table for which f(key)=key mod 13 is used with integer keys. Assuming linear probing is used for collision resolution, at which location would the key 103 be inserted, if the keys 661, 182, 24 and 103 are inserted in that order?

Answer

661 mod 13 = 11
182 mod 13 = 0
24 mod 13 = 11, already filled, so after linear probing it will get index 12 103 mod 13 = 12, already filled, so after linear probing it will get index 1



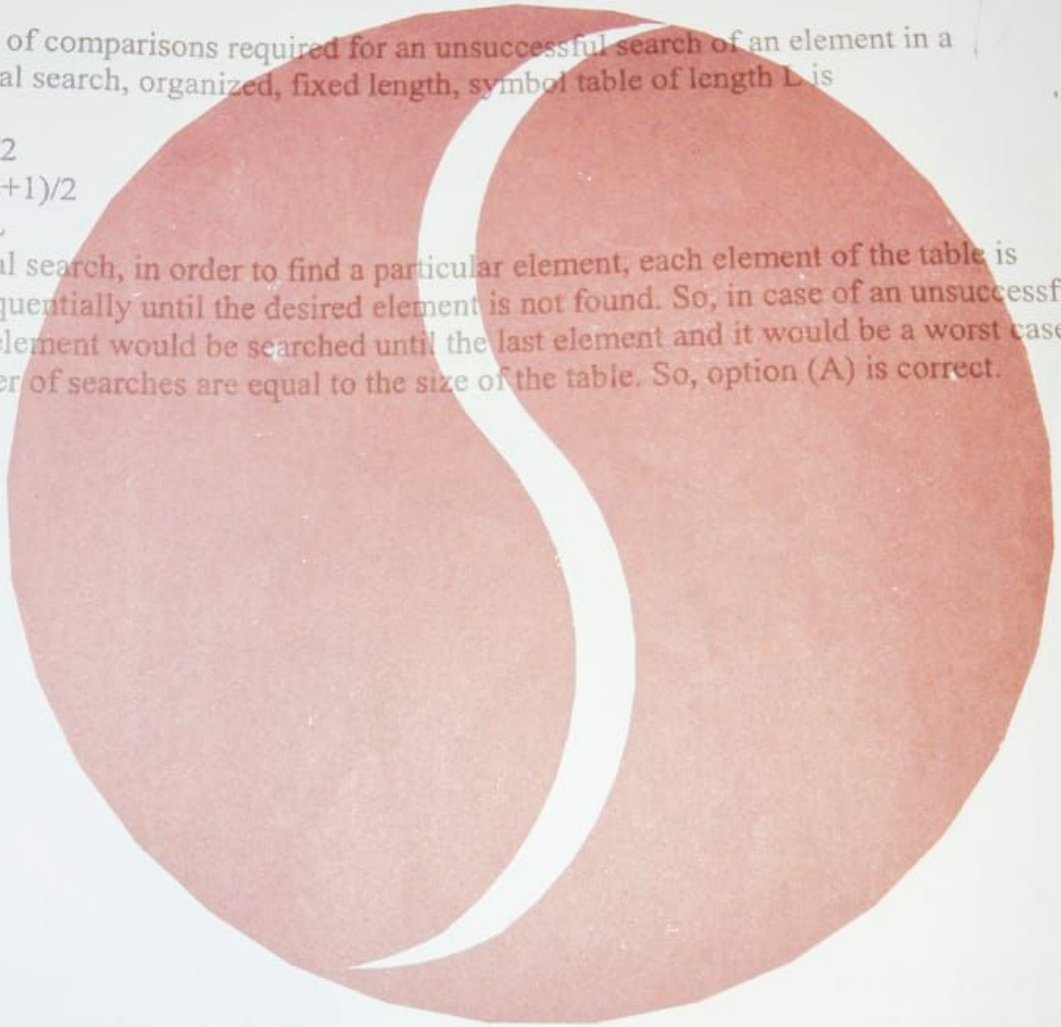
8. The average number of key comparisons done in a successful sequential search in a list of length it is

a) log n
b) (n-1)/2
c) n/2
d)(n+1)/2

9. Number of comparisons required for an unsuccessful search of an element in a sequential search, organized, fixed length, symbol table of length L is

a) L
b) L/2
c) (L+1)/2
d) 2L

In Sequential search, in order to find a particular element, each element of the table is searched sequentially until the desired element is not found. So, in case of an unsuccessful search, the element would be searched until the last element and it would be a worst case when number of searches are equal to the size of the table. So, option (A) is correct.



Final Exam Sample

Ministry of Higher Education
For Computer
Science and Management Technology
Department: Computer Science
Program: Computer Science
Time: 2 Hours
Examiners: Dr.

Year: 2018-2019
Term: 1st term
Subject: Data Structures
Code: C 201
Specialization: Computer Science–2nd ye
Aids: None
Total Marks: 75 marks

Number of Questions: 4 | Questions of Final Exam | Number of pages: 2

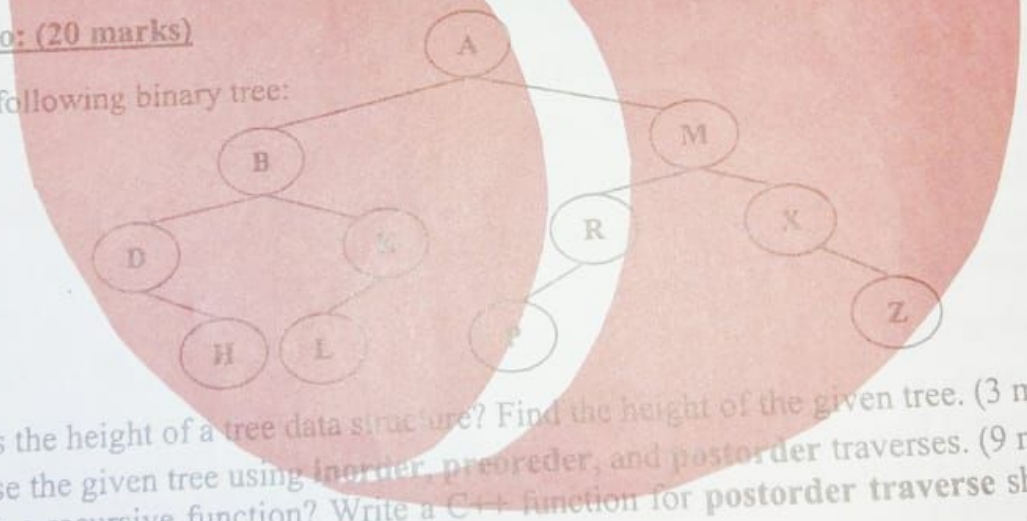
Answer the following questions:

Question One: (25 marks)

a- Apply the **selection sort** algorithm to sort the following array 4, 6, 2, 1, 8, 7, 3. (8 marks)
b- Write a C++ function to **pop an integer** from a **linked stack.** (8 marks)
c- Define collision? Store the values **14, 17, 25, 37, 34, 16, 70**using **linear probe**and **double hashingusing prime number = 5, array size = 11**. Show the situations of collision. (9 marks)

Question Two: (20 marks)

Consider the following binary tree:



a- What is the height of a tree data structure? Find the height of the given tree. (3 marks)
b- Traverse the given tree using **inorder, preoreder,** and **postorder** traverses. (9 marks)
c- What is a recursive function? Write a C++ function for **postorder traverse** showing the recursion instruction. (8 marks)

Question Three: (22 marks)

a- Consider a circular queue of size = 5, show the output of the following code segment: (5 marks)

```
void main( )
{enqueue (10);enqueue (20);
cout<<"Head = "<<head <<"Tail = "<<tail<<"\n";
enqueue (30);cout<<dequeue( )<<"\n";
int x = dequeue ( );
cout<<"Head = "<<head<<"Tail = "<<tail<<"\n";
}
```

b- Write a C++ function to **insert a new node** that contains a specific integer value and location entered as an input by the user in a **doubly linked list.** (7 marks)

c- Given a regular queue and a stack, both of size = N, use the functions *push, pop* and *dequeue*, reverse thecontents ofthat regular queue. (10 marks)

Question Four: Choose the correct answer : **(8 marks(one for each))**

1- What Member function places a new node at the end of the linked list?

a- insert_node( ) b- display_list ( ) c- new Node ( ) d- create_list ( )

2- A Binary Search Tree is traversed in the following order recursively: Right, root, left. The output sequence will be in:

a- Ascending order b- Descending order c- Bitomic sequence d- No specific order

3- The five items: A, B, C, D and E are pushed in stack, one after the other starting from A. The stack is popped four items and each element is inserted in a queue. Then two elements are deleted from the queue and pushed back on the stack. Now one item is popped from the stack. The popped item is:

a- A b- B c- C d- D

4- If a queue is implemented with a linked list, keeping track of a head node and a tail node with two reference variables. Which of these reference variables will change during an insertion into an EMPTY queue?

a- Only head changes b-Both change c-Only tail changes d- Neither changes

5- What are the number of nodes of left and right subtree of the binary tree if the data is inserted in the following order: 45, 15, 8, 56, 5, 65, 47, 12, 18, 10, 73, 50, 16, 61

a- 8 nodes to the left and 5 nodes to the right b- 6 nodes to the left and 7 nodes to the right
c- 7 nodes to the left and 6 nodes to the right d- 5 nodes to the left and 8 nodes to the right

6- With every popfrom the stack, the top:

a- Decrements by one b- Increments by one c- Stays as it is d-It is always zero

7- An overflow condition for a queue:

a- If (tail>size) b- If (tail<=head) c- If (tail<0) d- If (head>tail)

8- To insert an item into a stack, identify the correct set of statements:

a- top++; stack[top]=item; b- item = stack[top]; top --;
c- stack[item] = top; item++; d- stack[top] = item; top++;