Chapter Five

Chapter Name : Linked list

5.1 Linked List

```
#include <bits/stdc++.h>

using namespace std;


/* Link list node */

class Node {

public:

	int key;

	Node* next;

};


void push(Node** head_ref, int new_key)

{

	Node* new_node = new Node();

	new_node->key = new_key;

	new_node->next = (*head_ref);

	(*head_ref) = new_node;

}

bool search(Node* head, int x)

{

	Node* current = head; // Initialize current
```

```
        while (current != NULL) {
                if (current->key == x)
                        return true;
                current = current->next;
        }
        return false;
}
int main()
{
        Node* head = NULL;
        int x = 21;
        push(&head, 10);
        push(&head, 30);
        push(&head, 11);
        push(&head, 21);
        push(&head, 14);

        // Function call
        search(head, 21) ? cout << "Yes"<<21 : cout << "No";
        return 0;
}
```

Chapter Six

Chapter Name : Stack , Queue

6.1 Quicksort

(a) With numbers

```
#include<iostream>
using namespace std;
int part(int arr[],int s, int e)
{
   int pivot=arr[e];
   int index=s;
   for(int i=s;i<e;i++)
   {
      if(arr[i]<pivot)
            {
            int temp=arr[i];
            arr[i]=arr[index];
         arr[index]=temp;
            index++;


            }
   }
   int temp=arr[e];
```

```
   arr[e]=arr[index];
   arr[index]=temp;

   return index;
}
void quicksort(int arr[],int s,int e)
{
   if(s<e)
         {
         int p=part(arr,s,e);
         quicksort(arr,s,p-1);
      quicksort(arr,p+1,e);
   }
}
int main()
{
   int arr[100]={44,33,11,55,77,90,40,60,99,22,88,66};

   cout<<"Before sorting: \n";
   for(int i=0;i<12;i++)
   {
      cout<<arr[i]<<" ";
   }
   quicksort(arr,0,12);
```

```
  cout<<"after sorting: \n";

  for(int j=1;j<=12;j++)

  {

    cout<<arr[j]<<" ";

  }



}
```

```
"C:\New folder\akhdyfuhdkjh\quick.exe"
Before sorting:
44 33 11 55 77 90 40 60 99 22 88 66 after sorting:
11 22 33 40 44 55 60 66 77 88 90 99
Process returned 0 (0x0)   execution time : 2.285 s
Press any key to continue.
```

(b) With Alphabet

```
#include<iostream>


using namespace std;

int part(char arr[],char s, char e)

{

  char pivot=arr[e];

  char index=s;

  for(char i=s;i<e;i++)

  {
```

```
      if(arr[i]<pivot)
            {
            char temp=arr[i];
         arr[i]=arr[index];
         arr[index]=temp;
            index++;


            }
   }
   char temp=arr[e];
   arr[e]=arr[index];
   arr[index]=temp;
   return index;
}
void quicksort(char arr[],char s,char e)
{
   char p;
   if(s<e){
            p=part(arr,s,e);
      quicksort(arr,s,p-1);
      quicksort(arr,p+1,e);
  }
 }
int main()
```

```
{
   char arr[100]={'D','A','T','A','S','T','R','U','C','T','U','R','E'};
 cout<<"Before sorting: \n";
   for(char i=0;i<13;i++)
   {
      cout<<arr[i]<<" ";
   }
   quicksort(arr,0,5);
   cout<<endl<<"after sorting: \n";


for(char i=0;i<13;i++)
   {
          cout<<arr[i]<<" ";
   }
}
```

Outcome:

6.7

```
#include<iostream>

#define MAX 10
```

```
using namespace std;
int stack_arr[MAX];
int top=-1;
void push(int data)
{
   if(top==MAX-1)
   {
      cout<<"Stack overflow"<<endl;
   }
   top=top+1;
   stack_arr[top]=data;
}
int pop()
{
   int value;
   if(top==-1)
   {
      cout<<("Stack underflow");
            exit(1);
   }
   value=stack_arr[top];
   top=top-1;
   return value;
}
```

```
void display()
{
   if(top==-1)
   {
      cout<<"stack under flow"<<endl;
            return;
   }
   for(int i=top;i>=0;i--)
   cout<<stack_arr[i]<<" "<<endl;
}
int main()
{
   int data;
cout<<"Initial state:\n";
   push(11);
   push(22);
   push(33);



   for(int i=top;i>=0;i--)
   cout<<stack_arr[i]<<" "<<endl;
   data=pop();
   data=pop();
   data=pop();
```

```
    cout<<endl<<"After pop:\n";
    for(int i=top;i>=0;i--)
    cout<<stack_arr[i]<<" ";
    cout<<"\n";
    push(1);
    push(2);
    push(3);
    push(11);
    push(22);
    push(33);

    cout<<endl<<"Final state:\n";
            display();

    return 0;
}
```

Output:

```
C:\Users\shohe\OneDrive\Do

Initial state:
33
22
11

After pop:


Final state:
33
22
11
3
2
1
```

6.8 push and pop

```
#include<iostream>

#define MAX 5

using namespace std;

int stack_arr[MAX];

int top=-1;

void push(int data)

{

  if(top==MAX-1)

  {

    cout<<"Stack overflow"<<endl;
```

```
    }
    top=top+1;
    stack_arr[top]=data;
}
int pop()
{
    int value;
    if(top==-1)
    {
        cout<<("Stack underflow");
            exit(1);
    }
    value=stack_arr[top];
    top=top-1;
    return value;
}
void display()
{
    if(top==-1)
    {
        cout<<"stack under flow"<<endl;
            return;
    }
```

```
   for(int i=top;i>=0;i--)
   cout<<stack_arr[i]<<" "<<endl;
}
int main()
{
   int data;
   push(33);
   push(22);
   push(11);
   push(44);
   push(55);
   cout<<"PUSH 5 ELEMENTS:\n";
   for(int i=top;i>=0;i--)
   cout<<stack_arr[i]<<" "<<endl;
   cout<<"\n";
   cout<<"POP 2 ELEMENTS:\n";
   data=pop();
   data=pop();
         display();

   return 0;
}
```

Output:

```
C:\Users\shohe\OneDrive\Do

PUSH 5 ELEMENTS:
55
44
11
22
33

POP 2 ELEMENTS:
11
22
33
```

6.9 Queue

```
 #include <iostream>

using namespace std;

struct node {

   int data;

   struct node *next;

};

struct node* front = NULL;

struct node* rear = NULL;

struct node* temp;

void Insert() {

   int val;
```

```
  cout<<"Insert the element in queue : "<<endl;
  cin>>val;
  if (rear == NULL) {
        rear = (struct node *)malloc(sizeof(struct node));
        rear->next = NULL;
        rear->data = val;
        front = rear;
  } else {
        temp=(struct node *)malloc(sizeof(struct node));
        rear->next = temp;
        temp->data = val;
        temp->next = NULL;
        rear = temp;
  }
}
void Delete() {
  temp = front;
  if (front == NULL) {
    cout<<"Underflow"<<endl;
        return;
  }
  else
  if (temp->next != NULL) {
        temp = temp->next;
```

```
      cout<<"Element deleted from queue is : "<<front->data<<endl;
            free(front);
            front = temp;
   } else {
      cout<<"Element deleted from queue is : "<<front->data<<endl;
            free(front);
            front = NULL;
            rear = NULL;
   }
}
void Display() {
   temp = front;
   if ((front == NULL) && (rear == NULL)) {
      cout<<"Queue is empty"<<endl;
            return;
   }
   cout<<"Queue elements are: ";
   while (temp != NULL) {
      cout<<temp->data<<" ";
            temp = temp->next;
   }
   cout<<endl;
}
int main() {
```

```
  int ch;
  cout<<"1) Insert element to queue"<<endl;
  cout<<"2) Delete element from queue"<<endl;
  cout<<"3) Display all the elements of queue"<<endl;
  cout<<"4) Exit"<<endl;
  do {
    cout<<"Enter your choice : ";
        cin>>ch;
        switch (ch) {
        case 1: Insert();
        break;
        case 2: Delete();
        break;
        case 3: Display();
        break;
        case 4: cout<<"Exit"<<endl;
        break;
        default: cout<<"Invalid choice"<<endl;
        }
  } while(ch!=4);
  return 0;
}
```

Output

1) Insert element to queue

2) Delete element from queue

3) Display all the elements of queue

4) Exit

Enter your choice :

1

Insert the element in queue :

4

Enter your choice :

1

Insert the element in queue :

7

Enter your choice :

1

Insert the element in queue :

2

Enter your choice :

1

Insert the element in queue :

6

Enter your choice :

1

Insert the element in queue :

2

Enter your choice :

3

Queue elements are: 4 7 2 6 2

Enter your choice :

4

Exit

Chapter Seven

Chapter Name : Tree

7.1 Tree

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

struct Node {
  char data;
  struct Node *left, *right;
  Node(char data) {
    this->data = data;
    left = right = NULL;
  }
};

// Preorder traversal
void preorderTraversal(struct Node* node) {
  if (node == NULL)
    return;

  cout << node->data << "->";
  preorderTraversal(node->left);
  preorderTraversal(node->right);
}

// Postorder traversal
void postorderTraversal(struct Node* node) {
```

```
  if (node == NULL)
    return;

  postorderTraversal(node->left);
  postorderTraversal(node->right);
  cout << node->data << "->";
}

// Inorder traversal
void inorderTraversal(struct Node* node) {
  if (node == NULL)
    return;

  inorderTraversal(node->left);
  cout << node->data << "->";
  inorderTraversal(node->right);
}

int main() {
  struct Node* root = new Node('A');
  root->left = new Node('B');
  root->right = new Node('C');
  root->left->left = new Node('D');
  root->left->right = new Node('E');
  root->right->left = new Node('G');
  root->right->right = new Node('H');
  root->left->right->left = new Node('F');
  root->right->right->left = new Node('J');
  root->right->right->right = new Node('K');
  root->right->right->left->left= new Node('L');
  cout << "Inorder traversal ";
  inorderTraversal(root);

  cout << "\nPreorder traversal ";
  preorderTraversal(root);

  cout << "\nPostorder traversal ";
  postorderTraversal(root);
 return 0;

}
```

Output:

```
"C:\New folder\akhdyfuhdkjh\treeeee.exe"
Inorder traversal D->B->F->E->A->G->C->L->J->H->K->
Preorder traversal A->B->D->E->F->C->G->H->J->L->K->
Postorder traversal D->F->E->B->G->L->J->K->H->C->A->
Process returned 0 (0x0)   execution time : 9.415 s
Press any key to continue.
```

7.2 Terminal nodes of T

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

struct Node {
  char data;
  struct Node *left, *right;
  Node(char data) {
    this->data = data;
    left = right = NULL;
  }
};



void printLeafNodes(Node *root)
{

    if (!root)
        return;


    if (!root->left && !root->right)
    {
        cout << root->data << " ";
        return;
    }


    if (root->left)
```

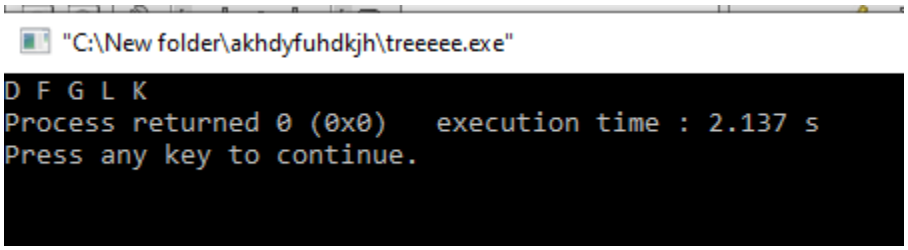
```
      printLeafNodes(root->left);


    if (root->right)
      printLeafNodes(root->right);
}

int main() {
  struct Node* root = new Node('A');
  root->left = new Node('B');
  root->right = new Node('C');
  root->left->left = new Node('D');
  root->left->right = new Node('E');
  root->right->left = new Node('G');
  root->right->right = new Node('H');
  root->left->right->left = new Node('F');
  root->right->right->left = new Node('J');
  root->right->right->right = new Node('K');
  root->right->right->left->left= new Node('L');

    printLeafNodes(root);
}
```

Output:



CHAPTER -8
GRAPH AND THEIR APPLICATION

8.1
(a)
Code:

```
#include <iostream>

using namespace std;

int adj[50][50];

int main()
{
    int M,N;
    cout<<"M is:";
    cin>>M;
    cout<<"N is:";
    cin>>N;

    int x,y;
    for(int i=1; i<=N; i++)
    {
        cin>>x>>y;
        adj[x][y] = 1;
        adj[y][x] = 1;
    }
    for(int i=1; i<=M; i++)
    {
        for(int j=1; j<=M; j++)
        {
            cout << adj[i][j] <<" ";
        }
        cout<<endl;
    }
    return 0;
}
```

Output:

```
M is:5
N is:8
3 4
5 3
2 4
1 5
3 2
4 2
3 1
5 1
0 0 1 0 1
0 0 1 1 0
1 1 0 1 1
0 1 1 0 0
1 0 1 0 0

Process returned 0 (0x0)   execution time : 31.151 s
Press any key to continue.
```