Name: Sanket Wakankar

Reg. No.: 2020BIT055

Sub: DAA

PRACTICAL NO. 1

Write C/C++ code to implement:

1.Stack:

```
// Concept of stack using vector.
// Sanket Wakankar (2020BIT055)
#include<bits/stdc++.h>
using namespace std;
class MyStack{
    public:
    vector<int> v;
    int sz = 0;
   void push(int x){
     v.push_back(x);
     sz++;
   }
   void pop(){
       v.pop_back();          // it is poped back.
       sz--;
   }
   int peek(){
       return v.back();
   }
   int size(){
    return sz;
   }
   bool is_empty(){
       return v.empty();
   }
};
int main(){
    MyStack s;
    s.push(5);
    s.push(10);
    s.push(20);
    s.pop();
    cout<<"Topmost element of stack is "<<s.peek()<<endl;
    if(s.is_empty()){
        cout<<"The stack is empty"<<endl;
```

```
    }
    else{
        cout<<"The stack is not empty"<<endl;
    }
    cout<<"Size of stack is "<<s.size()<<endl;
    return 0;
}
```

Output:

```
PS C:\Users\DELL\Documents\DSA Programming Using CPP> cd "c:\Users\DELL\Documents\DSA Programming Using CPP\.vscode\stack\" ; if ($?) { g++ tempCodeRunne
rFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
Topmost element of stack is 10
The stack is not empty
Size of stack is 2
PS C:\Users\DELL\Documents\DSA Programming Using CPP\.vscode\stack>
```

2.Queue:

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

int main() {

  // create a queue of int
  queue<int> nums;

  // push element into the queue
  nums.push(1);
  nums.push(2);
  nums.push(3);
  nums.pop();

  // get the element at the front
  int front = nums.front();
  cout << "First element in queue: " << front << endl;

  // get the element at the back
  int back = nums.back();
  cout << "Last element in queue: " << back << endl;

  return 0;
}
```

Output:

```
PS C:\Users\DELL\Documents\DSA Programming Using CPP> cd "c:\Users\DELL\Documents\DSA Programming Using CPP\.vscode\queue\" ; if ($?) { g++ tempCodeRunne
rFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
First element in queue: 2
Last element in queue: 3
PS C:\Users\DELL\Documents\DSA Programming Using CPP\.vscode\queue>
```

3. LinkedList:

```
// Sanket Wakankar (2020BIT055)
#include<iostream>
using namespace std;
class Node{
    public:
           int data;
           Node* next;
           Node(int x){
            data = x;
            next = NULL;
           }
};
Node* insert_in_sorted_linkedlist(Node* head,int element){
    Node* p = head;
    Node* a = new Node(element);
    if(p==NULL){
        a->next = NULL;
        return a;
    }
    else if(p->data>element){
        a->next = p;
        return a;
    }
    else
      while(p->next!=NULL && p->next->data<element){
        p = p->next;
    }
        a->next = p->next;
        p->next = a;


      return head;
}
void print_list(Node* head){
    while(head!=NULL){
        cout<<head->data<<" ";
        head = head->next;
    }
}

int main(){
    Node* head = new Node(13);
```

```
    Node* second = new Node(32);
    Node* third = new Node(36);
    Node* fourth = new Node(133);
    Node* fifth = new Node(180);
    head->next = second;
    second->next = third;
    third->next = fourth;
    fourth->next = fifth;
    fifth->next = NULL;

    Node* c = insert_in_sorted_linkedlist(head,150);
    print_list(c);
}
```

Output:

```
PS C:\Users\DELL\Documents\DSA Programming Using CPP> cd "c:\Users\DELL\Documents\DSA Programming Using
 ($?) { g++ Pract_everything2.cpp -o Pract_everything2 } ; if ($?) { .\Pract_everything2 }
13 32 36 133 150 180
PS C:\Users\DELL\Documents\DSA Programming Using CPP\.vscode\linked_list\singly linked_list>
```

4. Trees:

```
// Sanket Wakankar (2020BIT055)
// implementation using post order traversal
#include<bits/stdc++.h>
using namespace std;
class Node{
    public:  int key;
             Node* left;
             Node* right;
             Node(int x){
                key=x;
                left=NULL;
                right=NULL;
             }
};
void postorder(Node* root){
    if(root!=NULL){
        postorder(root->left);
        postorder(root->right);
        cout<<root->key<<" ";
    }
}
int main(){
    Node* root = new Node(10);
     root->left = new Node(20);
     root->right = new Node(30);
     root->right->left = new Node(40);
```

```
    root->right->right = new  Node(50);

    postorder(root);
    return 0;
}
```

Output:

```
PS C:\Users\DELL\Documents\DSA Programming Using CPP> cd "c:\Users\DELL\Documents\DSA
ersal.cpp -o postorder_traversal } ; if ($?) { .\postorder_traversal }
20 40 50 30 10
PS C:\Users\DELL\Documents\DSA Programming Using CPP\.vscode\Tree>
```

5. Graph

```
// Sanket Wakankar (2020BIT055)
// Representation of graph using STL
#include <bits/stdc++.h>
using namespace std;

void addEdge(vector<int> adj[], int u, int v)
{
    adj[u].push_back(v);
    adj[v].push_back(u);
}

// representation of graph
void printGraph(vector<int> adj[], int V)
{
    for (int v = 0; v < V; ++v) {
        cout << " Adjacency list of vertex " << v
             << "\n head ";
        for (auto x : adj[v])
            cout << "-> " << x;
        printf("\n");
    }
}

int main()
{
    int V = 5;
    vector<int> adj[V];
    addEdge(adj, 0, 1);
    addEdge(adj, 0, 4);
    addEdge(adj, 1, 2);
    addEdge(adj, 1, 3);
    addEdge(adj, 1, 4);
    addEdge(adj, 2, 3);
    addEdge(adj, 3, 4);
```

```
    printGraph(adj, V);
    return 0;
}
```

Output:

```
PS C:\Users\DELL\Documents\DSA Programming Using CPP> cd "c:\Users\DELL\Documents\DSA Programming
+ graph_representation.cpp -o graph_representation } ; if ($?) { .\graph_representation }
 Adjacency list of vertex 0
 head -> 1-> 4
 Adjacency list of vertex 1
 head -> 0-> 2-> 3-> 4
 Adjacency list of vertex 2
 head -> 1-> 3
 Adjacency list of vertex 3
 head -> 1-> 2-> 4
 Adjacency list of vertex 4
 head -> 0-> 1-> 3
PS C:\Users\DELL\Documents\DSA Programming Using CPP\.vscode\graph_data_structure>
```