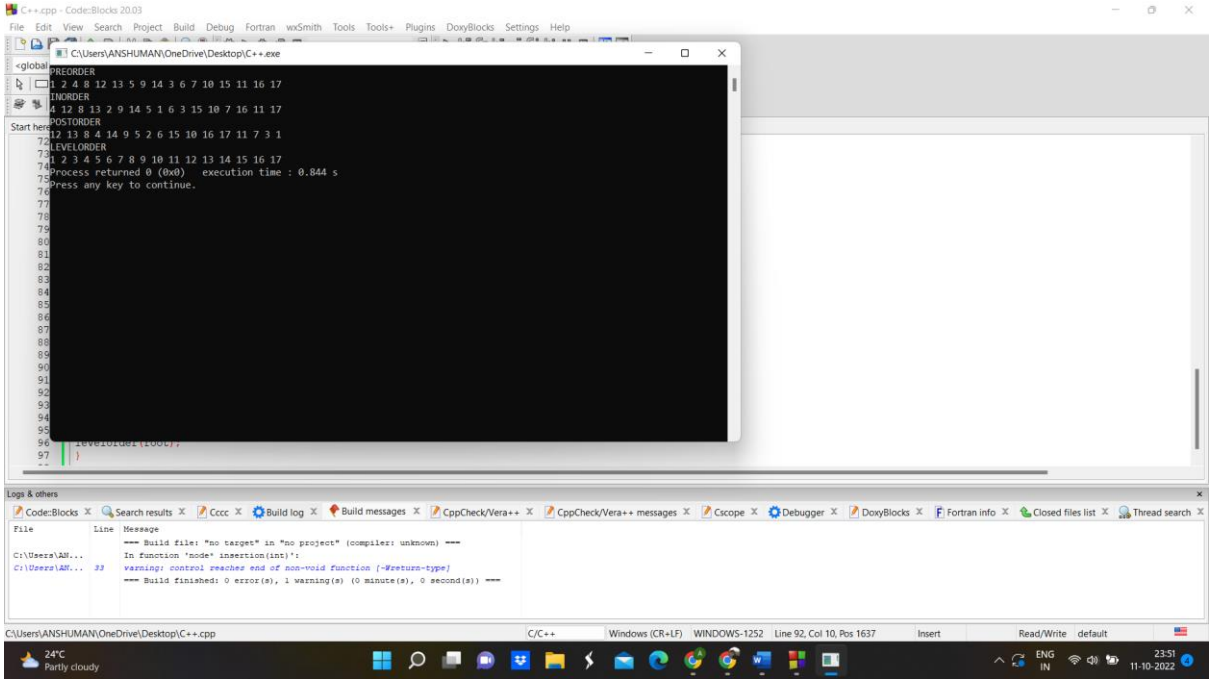
WEEK 9 – LAB-B

1)

```
#include <iostream>
#include<queue>
using namespace std;
struct node
{
    int data;
    node *left;
    node *right;
};
int a[]={1,2,3,4,5,6,7,0,8,9,0,0,0,10,11,12,13,12,13,0,14,0,0,0,0,0,0,15,0,16,17};
struct node* insertion(int n)
{
    node *temp=new node;
    temp=NULL;
    if(n>sizeof(a)/sizeof(int))
    {
        return temp;
    }
     else if(!a[n])
    {

        return temp;
    }
        else if(a[n])
        {
        temp=new node;
        temp->data=a[n];
        temp->left=insertion(2*n+1);
        temp->right=insertion(2*n+2);

    }

}
void postorder(node *root)
{
```

```
   if(root==NULL)
    return;
    else
    {
        postorder(root->left);
        postorder(root->right);
        cout<<root->data<<" ";
    }
}
void preorder(node *root)
{
    if(root==NULL)
    return;
    else
    {
        cout<<root->data<<" ";
        preorder(root->left);
        preorder(root->right);
    }
}
void inorder(node *root)
{
    if(root==NULL)
    return;
    else
    {
        inorder(root->left);
        cout<<root->data<<" ";
        inorder(root->right);
    }
}
void levelorder(node *root)
{
if (root == NULL) return;
  queue < node * > q;
  q.push(root);

  while (q.empty() == false) {
    node * node = q.front();
    cout << node -> data << " ";
    q.pop();
    if (node -> left != NULL)
      q.push(node -> left);
    if (node -> right != NULL)
      q.push(node -> right);
  }
}

int main() {
node *root;
root=insertion(0);
```

```
preorder(root);
cout<<endl;
inorder(root);
cout<<endl;
postorder(root);
cout<<endl;
levelorder(root);
}
```

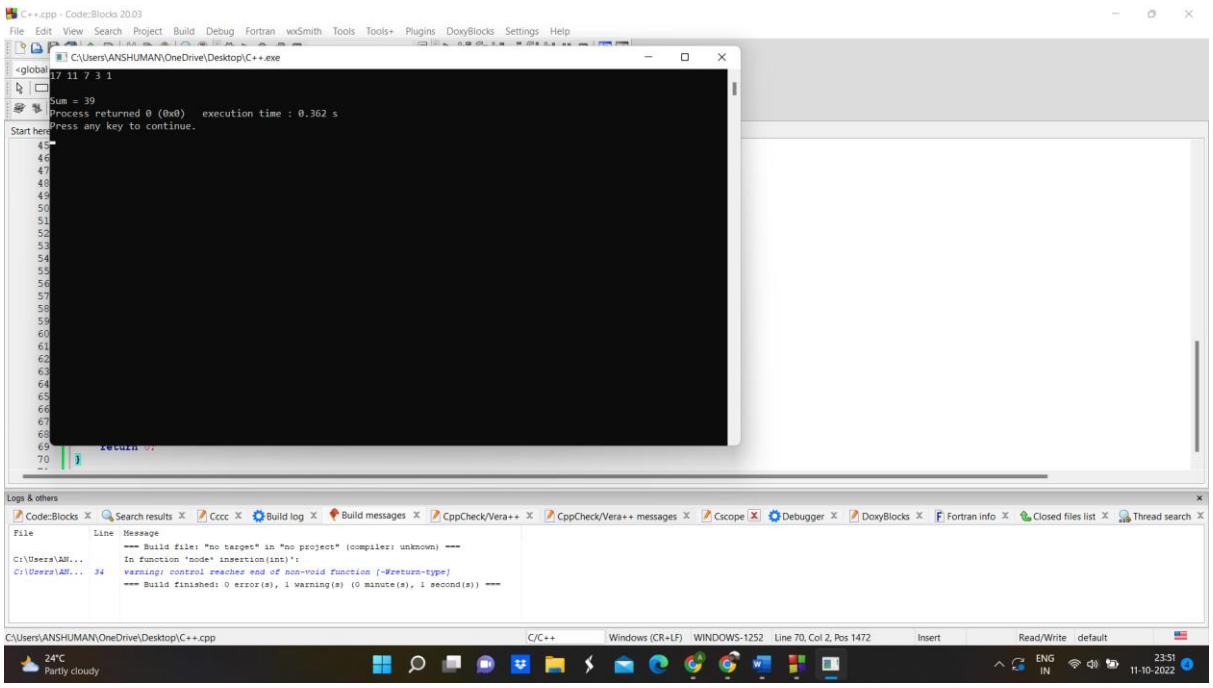


2)

```
#include <iostream>
#include<vector>
#include <bits/stdc++.h>
using namespace std;
struct node
{
   int data;
   node *left;
   node *right;
};
int maxsum=0;
int a[]={1,2,3,4,5,6,7,0,8,9,0,0,0,10,11,12,13,12,13,0,14,0,0,0,0,0,0,15,0,16,17};
struct node* insertion(int n)
{
   node *temp=new node;
   temp=NULL;
   if(n>sizeof(a)/sizeof(int))
   {
      return temp;
   }
    else if(!a[n])
   {

      return temp;
   }
      else if(a[n])
      {
      temp=new node;
```

```
        temp->data=a[n];
        temp->left=insertion(2*n+1);
        temp->right=insertion(2*n+2);

    }
}
vector<int> longestPath(node* root)
{

    if (root == NULL) {
        vector<int> temp
            = {};
        return temp;
    }

    vector<int> rightvect
        = longestPath(root->right);

    vector<int> leftvect
        = longestPath(root->left);

    if (leftvect.size() > rightvect.size())
        leftvect.push_back(root->data);

    else
        rightvect.push_back(root->data);

    return (accumulate(leftvect.begin(),leftvect.end(),0)
>accumulate(rightvect.begin(),rightvect.end(),0)
                ? leftvect
```

```
            : rightvect);
}
int main() {
node *root;
root=insertion(0);
vector<int> a=longestPath(root);
for (int i = 0; i < a.size(); i++)
        cout << a[i] << " ";
    cout << endl;
    cout << "\nSum = "
         << accumulate(a.begin(), a.end(), 0);
    return 0;
}
```

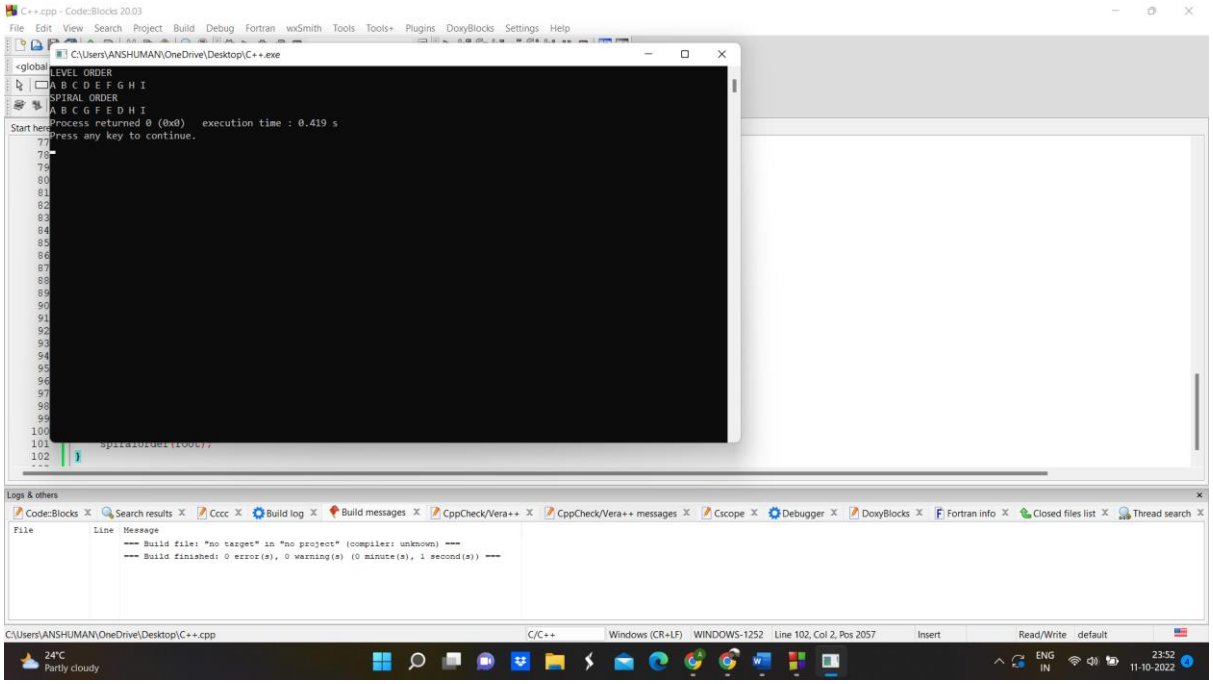


3)

```
#include <iostream>
#include<queue>
#include<stack>
#include <bits/stdc++.h>
using namespace std;
struct node
{
    char data;
    node *left;
    node *right;
};
struct node * newNode(char F)
{
    node *N=new node;
    N->left=NULL;
    N->right=NULL;
    N->data=F;
    return N;
}
node *insertnode(char c,queue<node *>&q,struct node *root)
{
    node * Node=newNode(c);
    if(root==NULL)
    {
     root=Node;
}
else if(q.front()->left==NULL)
    q.front()->left=Node;
else
{
    q.front()->right=Node;
    q.pop();
}
q.push(Node);
return root;
}
node * createtree()
{
    node *root=NULL;
    char arr[9]={'A','B','C','D','E','F','G','H','I'};
queue<node *>q;
for(int i=0;i<9;i++)
    root=insertnode(arr[i],q,root);
    return root;
}
void levelorder(node *root)
{
```

```
    if(root==NULL)
        return;
    queue<node *>q1;
    q1.push(root);
    while(q1.empty()==false)
    {
        node *f=q1.front();
        cout<<f->data<<" ";
        q1.pop();
        if(f->left!=NULL)
            q1.push(f->left);
        if(f->right!=NULL)
            q1.push(f->right);

    }
}
void spiralorder(node *root)
{
    stack<struct node*>s1;
    stack<struct node*>s2;
    s1.push(root);
    while(!s1.empty()||!s2.empty())
    {
        while(!s1.empty())
        {
            node *temp1=s1.top();
            cout<<temp1->data<<" ";
            s1.pop();
            if(temp1->right)
                s2.push(temp1->right);
            if(temp1->left)
                s2.push(temp1->left);
        }
        while(!s2.empty())
        {
            node *temp2=s2.top();
            cout<<temp2->data<<" ";
            s2.pop();
            if(temp2->left)
                s1.push(temp2->left);
            if(temp2->right)
                s1.push(temp2->right);
        }
    }
}
int main()
{
    node *root;
    root=createtree();
    cout<<"LEVEL ORDER"<<endl;
    levelorder(root);
```

```
    cout<<endl;
    cout<<"SPIRAL ORDER"<<endl;
    spiralorder(root);
}
```



4)

```
#include <iostream>
#include<queue>
using namespace std;
struct node
{
   int data;
   node *left;
   node *right;
};
int a[]={1,2,3,4,5,6,7,0,8,9,0,0,0,10,11,12,13,12,13,0,14,0,0,0,0,0,0,15,0,16,17};
struct node* insertion(int n)
{
   node *temp=new node;
   temp=NULL;
   if(n>sizeof(a)/sizeof(int))
   {
      return temp;
   }
    else if(!a[n])
   {

      return temp;
   }
      else if(a[n])
      {
      temp=new node;
      temp->data=a[n];
      temp->left=insertion(2*n+1);
```

```
        temp->right=insertion(2*n+2);

    }

}
void postorder(node *root)
{
   if(root==NULL)
    return;
    else
    {
        postorder(root->left);
        postorder(root->right);
        cout<<root->data<<" ";
    }
}
void preorder(node *root)
{
    if(root==NULL)
    return;
    else
    {
        cout<<root->data<<" ";
        preorder(root->left);
        preorder(root->right);
    }
}
void inorder(node *root)
{
```

```
    if(root==NULL)
    return;
    else
    {
        inorder(root->left);
        cout<<root->data<<" ";
        inorder(root->right);
    }
}
void levelorder(node *root)
{
if (root == NULL) return;
  queue < node * > q;
  q.push(root);

  while (q.empty() == false) {
    node * node = q.front();
    cout << node -> data << " ";
    q.pop();
    if (node -> left != NULL)
      q.push(node -> left);
    if (node -> right != NULL)
      q.push(node -> right);
  }
}

int main() {
node *root;
root=insertion(0);
```

```
preorder(root);
cout<<endl;
inorder(root);
cout<<endl;
postorder(root);
cout<<endl;
levelorder(root);
}
```