Assignment No: 6

Aim:

A book consists of chapters consists of sections of subsections construct a tree and print the nodes. Find time and space requirements of our method.

Theory:

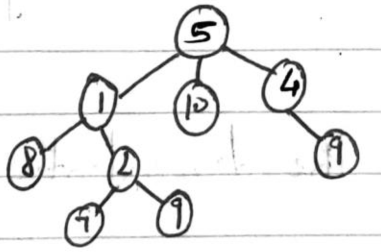
Introduction to Tree ⇒

Defn:-

A tree is a set of node storing elements such that nodes have a parent-child releationship that satisfies

if T is not empty, T has a special tree called root that has no parent.

each node v of T different than the root has a unique parent node w:



An internal/inner node is any node of tree that has child node.

There are two basic types of trees. In unordered tree, a tree in a purely structural sense that is to say, there is no order for children of that node, ordered trees are by far the most common form of tree data structure.

(i) Path ⇒

It refers to sequence of nodes along edge of tree.

(ii) Root ⇒

The node at top of tree is called root.

(iii) Parent ⇒ ~~A~~

Any node except that root node has one edge upword to a node called parent.

(iv) Child ⇒

The node below a given node is connected by its edge downward is child node.

(v) Leaf ⇒

Node which doesnot have any child node.

(vi) Subtree ⇒

It represents the descendants of a node.

(vii) Visiting ⇒

It refers to checking the value of node when control is on node.

(viii) Traversing ⇒

It means passing through nodes in specific order.

(ix) Levels ⇒

It represents generation of node. If root node is at level 0, then its next child node level is at 1.

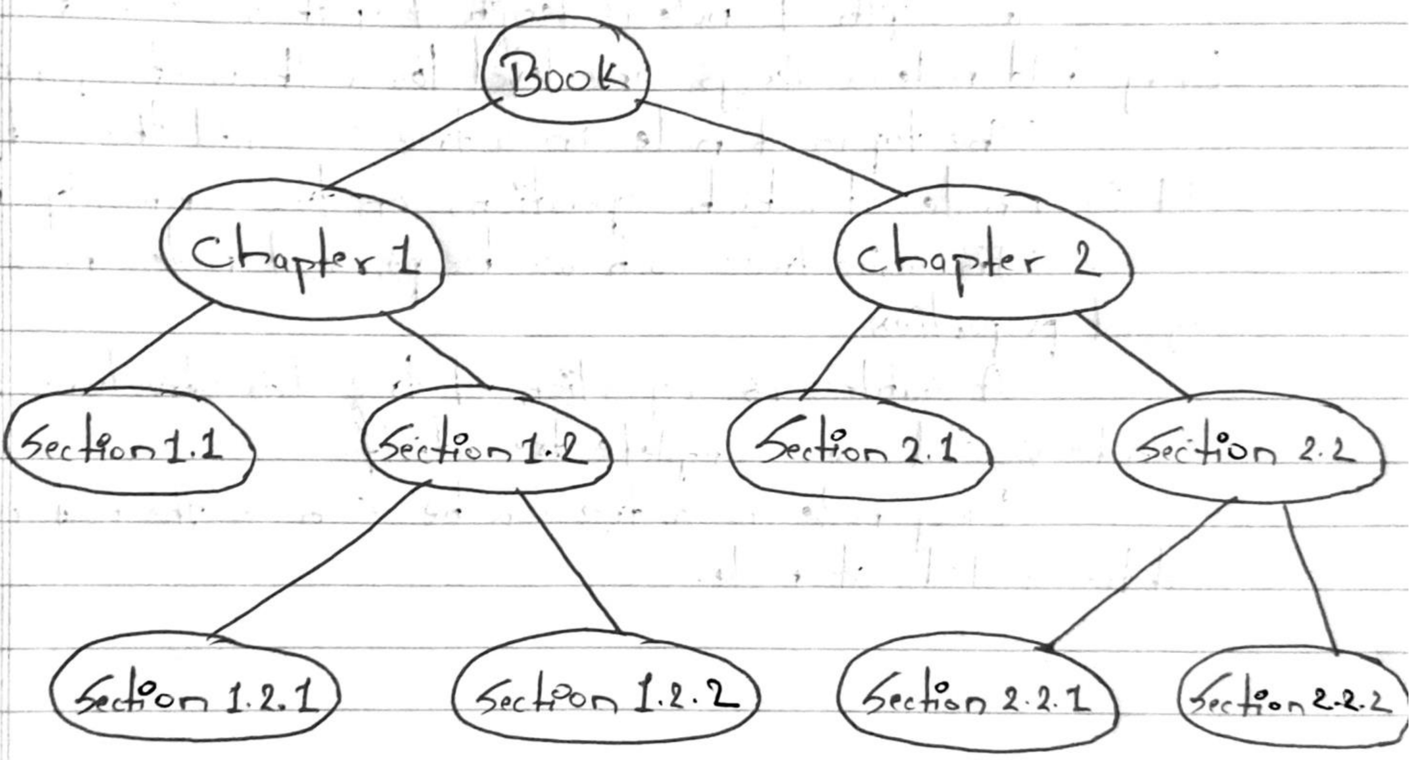
Advantages of Tree ⇒

Tree reflects structural relationship in data.

Tree are used to represent hierarchies.

Tree provide an efficient insertion & searching

Tree are flexible data, allowing to move subtrees around with minimum effort.



Recursive Definition ⇒

T is either empty, or consist of node r & possibily empty set of trees whose roots are children for r.

Trees is widely used data structure with a set of linked-~~list~~ nodes, it graphically represents most commonly.

Data are stored in nodes, from which tree is consisted of.

A node that has a child is called child's parent node A node has at most one parent. The height of root is height of tree. In other words "height" of tree is "number of level" in tree.

Height of tree with no elements is 0.

Height of tree with 1 element is 0

Height of tree with >1 element is equal to it height of all its tallest subtree.

The depth of node is length of path to its root. Every child node is always one level lower than its parent.

The top most node in a tree is called the root node. It is a node at which operations on tree commonly begin.

In some trees such as heaps, root nodes has special proporties.

A subtree is a portion of tree data structure that can be viewed as complete tree itself.

Every node in a tree can be seen as the root node of subtree rooted at node.

Program Code:-

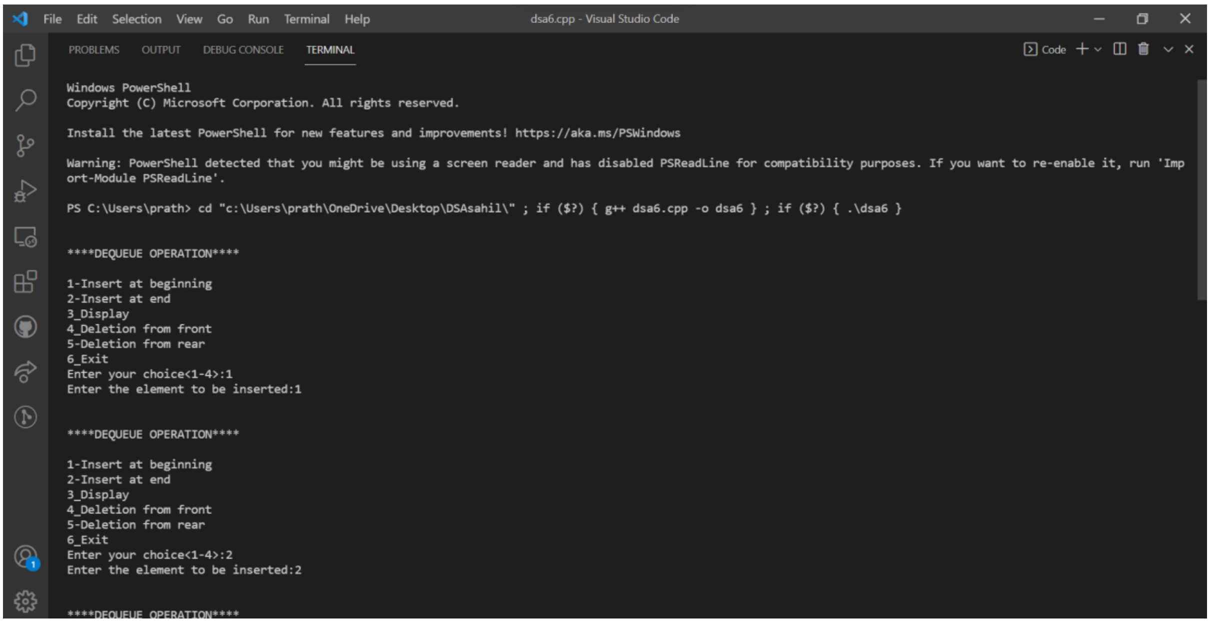
```
//Program to illustrate double ended queue using array.
#include<iostream>
using namespace std;
#define SIZE 5
class dequeue
{
        int a[10],front,rear,count;
public:
        dequeue();
        void add_at_beg(int);
        void add_at_end(int);
        void delete_fr_front();
        void delete_fr_rear();
        void display();
};


dequeue::dequeue()
{
        front=-1;
        rear=-1;
        count=0;
}


void dequeue::add_at_beg(int item)
{
        int  i;
        if(front==-1)
        {
                front++;
                rear++;
                a[rear]=item;
                count++;
        }
        else if(rear>=SIZE-1)
        {
                cout<<"\nInsertion is not possible,overflow!!!!";
        }
        else
        {
                for(i=count;i>=0;i--)
                {
                        a[i]=a[i-1];
                }
                a[i]=item;
                count++;
                rear++;
        }
}



void dequeue::add_at_end(int item)
{

        if(front==-1)
        {
                front++;
                rear++;
                a[rear]=item;
                count++;
        }
        else if(rear>=SIZE-1)
        {
                cout<<"\nInsertion is not possible,overflow!!!";
                return;
        }
        else
        {
                a[++rear]=item;
        }
}


void dequeue::display()
{

        for(int i=front;i<=rear;i++)
        {
                cout<<a[i]<<" ";        }
}


void dequeue::delete_fr_front()
{
        if(front==-1)
        {
```

```
		cout<<"Deletion is not possible:: Dequeue is empty";
		return;
	}
	else
	{
		if(front==rear)
		{
			front=rear=-1;
			return;
		}
		cout<<"The deleted element is "<<a[front];
		front=front+1;
	}
}

void dequeue::delete_fr_rear()
{
	if(front==-1)
	{
		cout<<"Deletion is not possible:Dequeue is empty";
		return;
	}
	else
	{
		if(front==rear)
		{
			front=rear=-1;
		}
		cout<<"The deleted element is "<< a[rear];
		rear=rear-1;
```

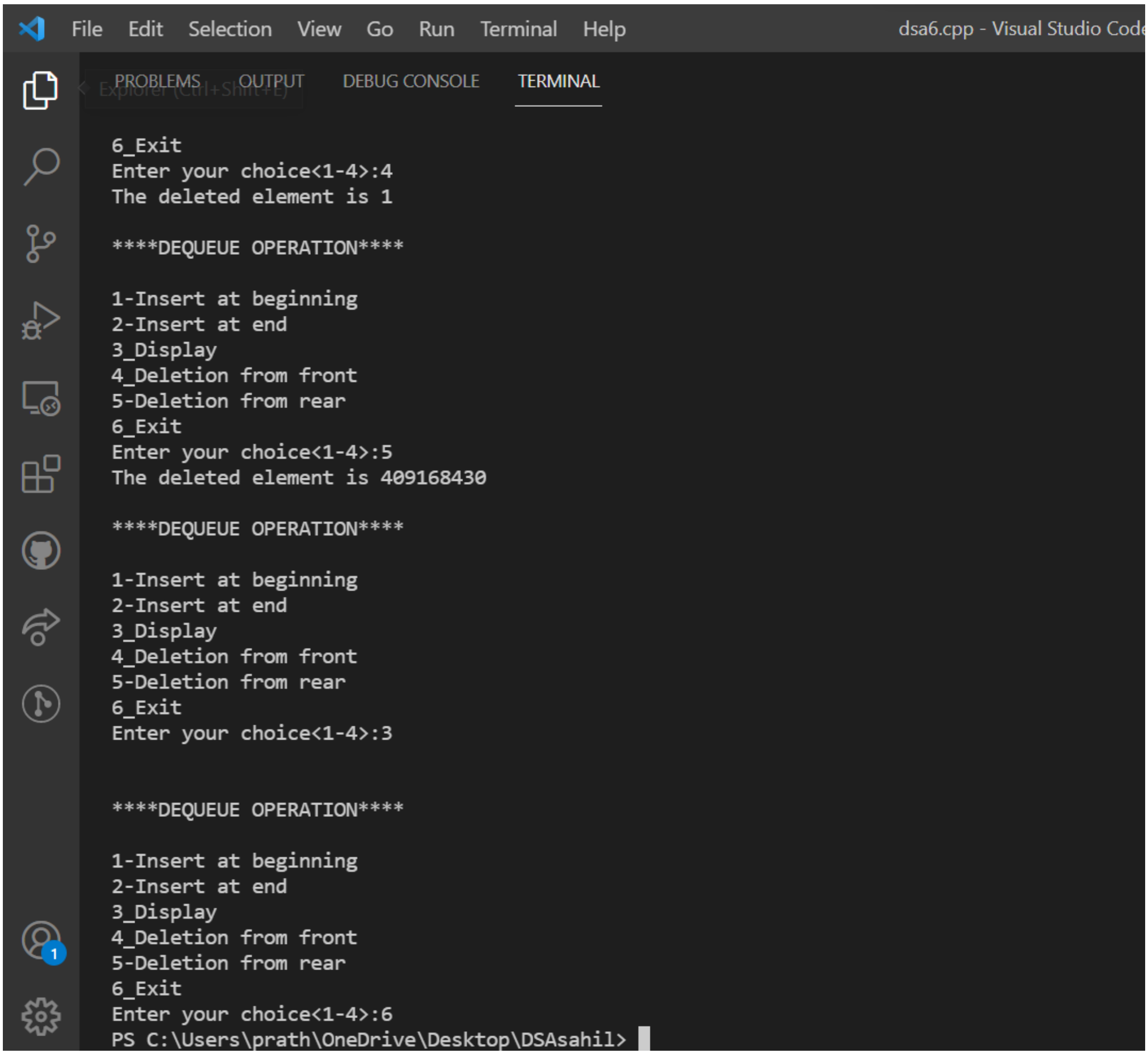
```
          }
}



int main()
{
          int c,item;
          dequeue d1;

          do
          {
                    cout<<"\n\n****DEQUEUE OPERATION****\n";
                    cout<<"\n1-Insert at beginning";
                    cout<<"\n2-Insert at end";
                    cout<<"\n3_Display";
                    cout<<"\n4_Deletion from front";
                    cout<<"\n5-Deletion from rear";
                    cout<<"\n6_Exit";
                    cout<<"\nEnter your choice<1-4>:";
                    cin>>c;

                    switch(c)
                    {
                    case 1:
                              cout<<"Enter the element to be inserted:";
                              cin>>item;
                              d1.add_at_beg(item);
                              break;

                    case 2:
```

```
            cout<<"Enter the element to be inserted:";
            cin>>item;
            d1.add_at_end(item);
            break;

        case 3:
            d1.display();
            break;

        case 4:
            d1.delete_fr_front();
            break;
        case 5:
            d1.delete_fr_rear();
            break;

        case 6:
            exit(1);
            break;

        default:
            cout<<"Invalid choice";
            break;
        }

    }while(c!=7);
    return 0;

}
```

Program Output :-


File Edit Selection View Go Run Terminal Help
dsa6.cpp - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Code

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to re-enable it, run 'Imp
ort-Module PSReadLine'.

PS C:\Users\prath> cd "c:\Users\prath\OneDrive\Desktop\DSAsahil\" ; if ($?) { g++ dsa6.cpp -o dsa6 } ; if ($?) { .\dsa6 }


****DEQUEUE OPERATION****

1-Insert at beginning
2-Insert at end
3_Display
4_Deletion from front
5-Deletion from rear
6_Exit
Enter your choice<1-4>:1
Enter the element to be inserted:1


****DEQUEUE OPERATION****

1-Insert at beginning
2-Insert at end
3_Display
4_Deletion from front
5-Deletion from rear
6_Exit
Enter your choice<1-4>:2
Enter the element to be inserted:2


****DEQUEUE OPERATION****
```


File Edit Selection View Go Run Terminal Help
dsa6.cpp - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Explorer (Ctrl+Shift+E)

```
6_Exit
Enter your choice<1-4>:4
The deleted element is 1

****DEQUEUE OPERATION****

1-Insert at beginning
2-Insert at end
3_Display
4_Deletion from front
5-Deletion from rear
6_Exit
Enter your choice<1-4>:5
The deleted element is 409168430

****DEQUEUE OPERATION****

1-Insert at beginning
2-Insert at end
3_Display
4_Deletion from front
5-Deletion from rear
6_Exit
Enter your choice<1-4>:3


****DEQUEUE OPERATION****

1-Insert at beginning
2-Insert at end
3_Display
4_Deletion from front
5-Deletion from rear
6_Exit
Enter your choice<1-4>:6
PS C:\Users\prath\OneDrive\Desktop\DSAsahil>
```