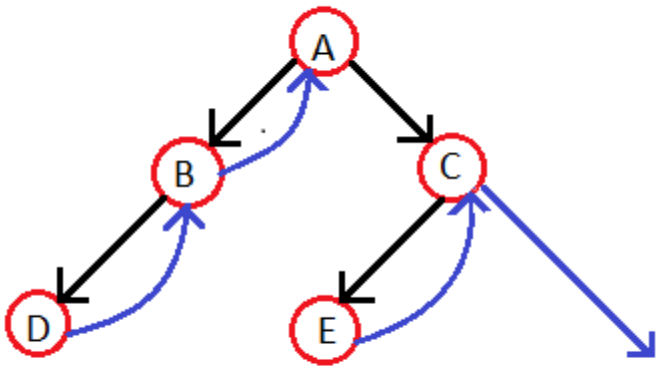
ASSIGNMENT 2

AIM:- Construct a threaded binary search tree by inserting value in the given order and traverse it in inorder traversal using threads.

OBJECTIVE:- Traverse a threaded binary search tree

THEORY:- Inorder traversal of a Binary tree can either be done using recursion or with the use of a auxiliary stack. The idea of threaded binary trees is to make inorder traversal faster and do it without stack and without recursion. A binary tree is made threaded by making all right child pointers that would normally be NULL point to the inorder successor of the node (if it exists).
There are two types of threaded binary trees.
Single Threaded: Where a NULL right pointers is made to point to the inorder successor (if successor exists)
Double Threaded: Where both left and right NULL pointers are made to point to inorder predecessor and inorder successor respectively. The predecessor threads are useful for reverse inorder traversal and postorder traversal.
The threads are also useful for fast accessing ancestors of a node.



ALGORITHM:- void inOrder(struct Node *root)

```
{
   struct Node *cur = leftmost(root);
   while (cur != NULL)
   {
      printf("%d", cur->data);

      // If this node is a thread node, then go to
      // inorder successor
      if (cur->rightThread)
         cur = cur->right;
      else // Else go to the leftmost child in right subtree
         cur = leftmost(cur->right);
   }
}
```

CODE:-

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

struct treeNode
{
```

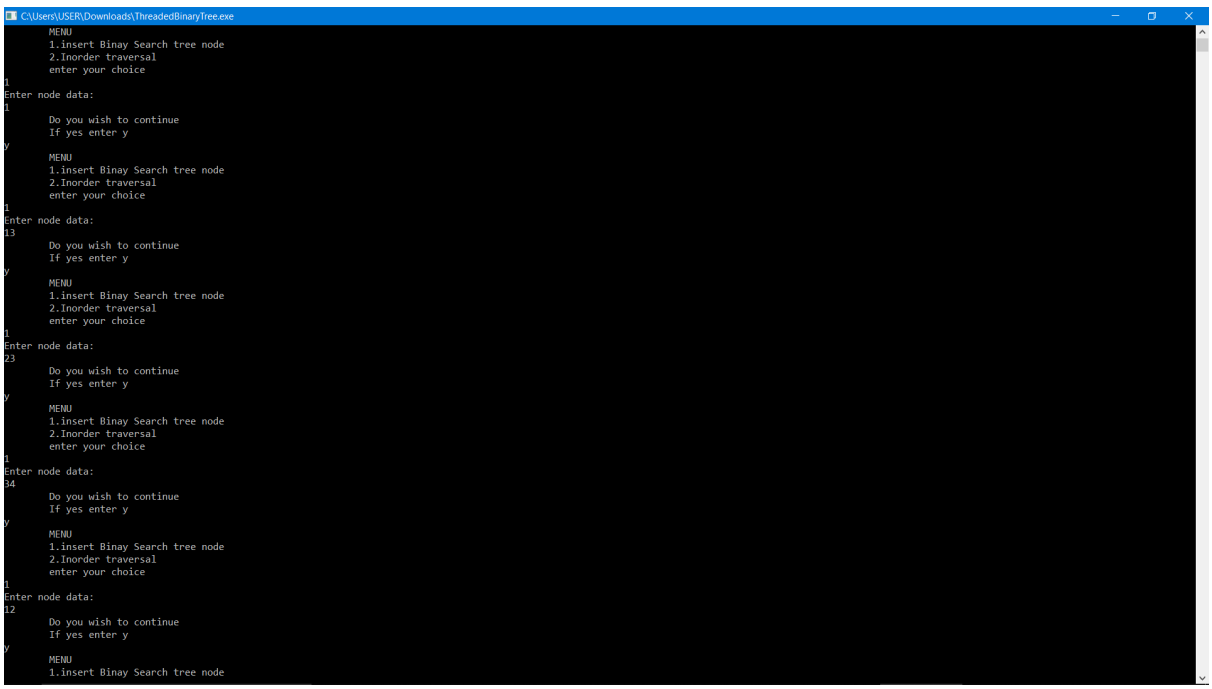
```
        int data;
        treeNode* left;
        treeNode* right;
        bool lthread;
        bool rthread;
};

class Tree
{
        private :
                treeNode* headnode = NULL;
        public :
                Tree()
                {
                        headnode = new treeNode();
                        headnode->lthread = false;
                        headnode->rthread = true;
                        headnode->left = headnode->right = headnode;
                }
                void insert(int data)
                {
                        treeNode* nn = new treeNode;
                        nn->left = NULL;
                        nn->right = NULL;
                        nn->data = data;
                        if(headnode == headnode->left && headnode ==
headnode->right)
                        {
                                nn->left = headnode;
                                headnode->left = nn;
                                nn->lthread = headnode->lthread;
                                headnode->lthread = true;
                                nn->right = headnode;
                        }

                        else
                        {
                                treeNode* current = headnode->left;
                                while(true)
                                {
```

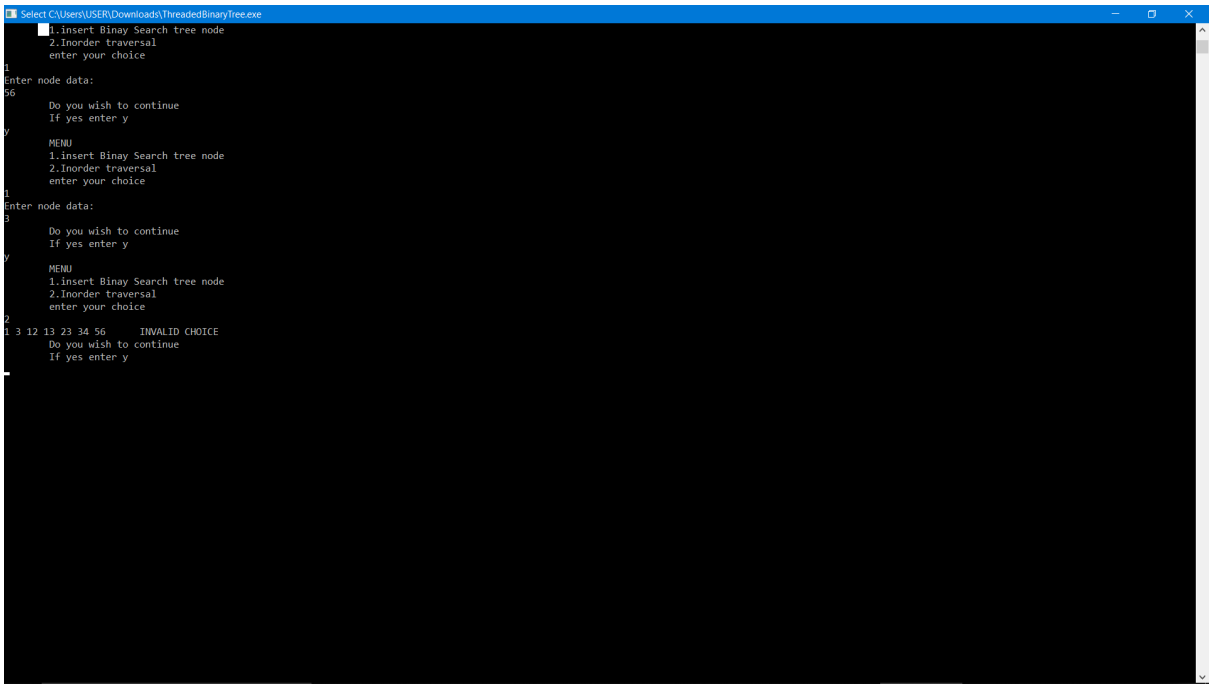
```
                        if(current->data > nn->data)
                        {
                                if(current->lthread == false)
                                {
                                        nn->left = current->left;
                                        current->left = nn;
                                        nn->lthread = current->lthread;
                                        current->lthread = true;
                                        nn->right = current;
                                        break;
                                }

                                else
                                        current = current->left;
                        }

                        else
                        {
                                if(current->rthread == false)
                                {
                                        nn->right = current->right;
                                        current->right = nn;
                                        nn->rthread = current
->rthread;
                                        current->rthread = true;
                                        nn->left = current;
                                        break;
                                }
                                else
                                {
                                        current = current->right;
                                }
                        }

                }

        }

}
```

```
        void inorder()
        {
                treeNode* current = headnode->left;
                while (current->lthread == true)
                        current = current->left;

                while (current != headnode)
                {
                        cout << current->data<<" ";
                        if(current->rthread == false)
                        {
                                current = current->right;
                        }
                        else
                        {
                                current = current->right;
                                while (current->lthread != false)
                                current = current->left;
                        }


                }
        }

        treeNode* getRoot()
        {
                return headnode->left;
        }
}tree;


int main()
{
        char choice='y';
        int ch;
        int data;

        while(choice == 'y')
        {
                cout << "\tMENU" << endl;
```

```
            cout << "\t1.insert Binay Search tree node" <<  endl;
            cout << "\t2.Inorder traversal" << endl;
            cout << "\tenter your choice" << endl;
            cin >> ch;
            switch(ch)
            {
                  case 1:
                        cout << "Enter node data: " << endl;
                        cin >> data;
                        tree.insert(data);
                        break;
                  case 2 :
                        tree.inorder();
                        break;
                  default :
                        cout << "\tINVALID CHOICE" << endl;

            }
            cout << "\tDo you wish to continue" << endl;
            cout << "\tIf yes enter y" << endl;
            cin >> choice;
      }
}
```

OUTPUT:-





CONCLUSION:- Threaded trees make in- order tree traversal a little faster, because you you have guaranteed O(1) time to access the next node. This is opposed to a regular binary tree where it would take O(lg n) time, because you have to "climb" up the tree and then back down.