Rajalakshmi Engineering College

Name: Naveed Sheriff
Email: 240701348@rajalakshmi.edu.in
Roll no: 240701348
Phone: 9025573780
Branch: REC
Department: I CSE FD
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_CY_Updated

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

Dhruv is working on a project where he needs to implement a Binary Search Tree (BST) data structure and perform various operations on it.

He wants to create a program that allows him to build a BST, traverse it in different orders (inorder, preorder, postorder), and exit the program when needed.

Help Dhruv by designing a program that fulfils his requirements.

Input Format

The first input consists of the choice.

If the choice is 1, enter the number of elements N and the elements inserted into

the tree, separated by a space in a new line.

If the choice is 2, print the in-order traversal.

If the choice is 3, print the pre-order traversal.

If the choice is 4, print the post-order traversal.

If the choice is 5, exit.

Output Format

The output prints the results based on the choice.

For choice 1, print "BST with N nodes is ready to use" where N is the number of nodes inserted.

For choice 2, print the in-order traversal of the BST.

For choice 3, print the pre-order traversal of the BST.

For choice 4, print the post-order traversal of the BST.

For choice 5, the program exits.

If the choice is greater than 5, print "Wrong choice".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1
5
12 78 96 34 55
2
3
4
5
Output: BST with 5 nodes is ready to use
BST Traversal in INORDER
12 34 55 78 96

BST Traversal in PREORDER
12 78 34 55 96
BST Traversal in POSTORDER
55 34 96 78 12

Answer

```
#include<stdio.h>
#include<stdlib.h>

struct Node
{
   int data;
   struct Node*left;
   struct Node*right;
};

struct Node*insert(struct Node*root , int e)
{
   struct Node*newnode = (struct Node*)malloc(sizeof(struct Node));
   if(root == NULL)
   {
      newnode->data = e;
      newnode->left = NULL;
      newnode->right = NULL;
      root = newnode;
   }
   else if(e < root->data)
      root->left = insert(root->left , e);
   else if(e > root->data)
      root->right = insert(root->right , e);
   return root;
}

void inorder(struct Node*root)
{
   if(root == NULL)
      return;
   inorder(root->left);
   printf("%d ",root->data);
   inorder(root->right);

}
```

```
void preorder(struct Node*root)
{
   if(root == NULL)
      return;
   printf("%d ",root->data);
   preorder(root->left);
   preorder(root->right);

}

void postorder(struct Node*root)
{
   if(root == NULL)
      return;

   postorder(root->left);
   postorder(root->right);
   printf("%d ",root->data);

}

int main()
{
   struct Node*root = NULL;
   int ch;
   while(1)
   {
      scanf("%d",&ch);
      switch(ch)
      {
         case 1:
         {
         root = NULL;
         int n;
         scanf("%d",&n);
         printf("BST with %d nodes is ready to use\n",n);
         for(int i=0;i<n; i++)
         {
            int e;
            scanf("%d",&e);
            root = insert(root,e);
```

```
        }
        break;
        }

        case 2:{
        printf("BST Traversal in INORDER\n");
        inorder(root);
        printf("\n");
        break;
        }

        case 3:
        {
        printf("BST Traversal in PREORDER\n");
        preorder(root);
        printf("\n");
        break;
        }

        case 4:{
        printf("BST Traversal in POSTORDER\n");
        postorder(root);
        printf("\n");
        break;
        }

        case 5:
        {
        return 0;
        }

        default:{
        printf("Wrong choice\n");
        break;
        }
    }
  }
}
```

Status : Correct ***Marks : 10/10***

2. Problem Statement

Emily is studying binary search trees (BST). She wants to write a program that inserts characters into a BST and then finds and prints the minimum and maximum values.

Guide her with the program.

Input Format

The first line of input consists of an integer N, representing the number of values to be inserted into the BST.

The second line consists of N space-separated characters.

Output Format

The first line of output prints "Minimum value: " followed by the minimum value of the given inputs.

The second line prints "Maximum value: " followed by the maximum value of the given inputs.

Refer to the sample outputs for formatting specifications.

Sample Test Case

Input: 5
Z E W T Y

Output: Minimum value: E
Maximum value: Z

Answer

```
#include<stdio.h>
#include<stdlib.h>

struct node
{
   char data;
   struct node*left;
   struct node*right;
```

```
};
struct node*insert(struct node*root , char e)
{
   struct node*newnode = (struct node*)malloc(sizeof(struct node));
   if(root == NULL)
   {
      newnode->data = e;
      newnode->left = NULL;
      newnode->right = NULL;
      root = newnode;
   }
   else if(e < root->data)
      root->left = insert(root->left , e);
   else if(e > root->data)
      root->right = insert(root->right , e);
   return root;
}

char findmin(struct node*root)
{
   if(root != NULL)
   {
      while(root->left != NULL)
         root = root->left;
   }
   return root->data;
}

char findmax(struct node*root)
{
   if(root != NULL)
   {
      while(root->right != NULL)
         root = root->right;
   }
   return root->data;
}

int main()
{
   struct node*root = NULL;
```

```
    int n;
    char c;
    scanf("%d",&n);
    for(int i=0; i<n;i++)
    {
        scanf(" %c ",&c);
        root = insert(root,c);
    }
    char min,max;
    min = findmin(root);
    max = findmax(root);
    printf("Minimum value: %c\n",min);
    printf("Maximum value: %c",max);
}
```

Status : Correct **Marks : 10/10**

3. Problem Statement

Edward has a Binary Search Tree (BST) and needs to find the k-th largest element in it.

Given the root of the BST and an integer k, help Edward determine the k-th largest element in the tree. If k exceeds the number of nodes in the BST, return an appropriate message.

Input Format

The first line of input consists of integer n, the number of nodes in the BST.

The second line consists of the n elements, separated by space.

The third line consists of the value of k.

Output Format

The output prints the kth largest element in the binary search tree.

For invalid inputs, print "Invalid value of k".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 7
8 4 12 2 6 10 14
1

Output: 14

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>

struct node
{
   int data;
   struct node*left;
   struct node*right;
};

struct node*insert(struct node*root , int e)
{
   struct node*newnode = (struct node*)malloc(sizeof(struct node));
   if(root == NULL)
   {
      newnode->data = e;
      newnode->left = NULL;
      newnode->right = NULL;
      root = newnode;
   }
   else if(e < root->data)
   root->left = insert(root->left , e);
   else if(e > root->data)
   root->right = insert(root->right , e);
   return root;
}

void findkthmax(struct node*root , int k, int*count , int*result)
{
   if(root == NULL || *count >=k)
      return;
```

```
    findkthmax(root->right , k,count, result);

    (*count)++;
    if(*count == k){
        *result = root->data;
        return;
    }
    findkthmax(root->left , k ,count, result);
}


int main()
{
    struct node*root = NULL;
    int n,e;
    scanf("%d",&n);
    for(int i=0;i<n; i++)
    {
        scanf("%d",&e);
        root = insert(root , e);
    }
    int k;
    scanf("%d",&k);
    int count = 0,result = -1;
    findkthmax(root , k,&count,&result);
    if(count < k || k <= 0)
    printf("Invalid value of k\n");
    else
    printf("%d\n",result);
    return 0;
}
```

Status : Correct **Marks : 10/10**