6)Binary tree construction and Tree traversal Operation

```
#include<stdio.h>

#include<malloc.h>

#include<string.h>

#include<stdlib.h>

#define MAX 30


struct tree {

  int info;

  struct tree* left;

  struct tree* right;

};


#define MALLOC(p, s, t) \

  p = (t)malloc(s); \

  if (p == NULL) { \

    printf("insufficient memory\n"); \

    exit(0); \

  }


typedef struct tree* NODE;


NODE create(NODE, int);

NODE createtree(NODE, int);

void Preorder(NODE);

void Postorder(NODE);
void Inorder(NODE);
int search(NODE, int);

int n;

int main() {
    int choice, done, flag, key;
    NODE p;
    p = NULL;
    done = 0;
    while (!done) {
        printf("1.Create\t2.Preorder\t3.Inorder\t4.Postorder\t5.Search\t6.Exit\n");
        printf("Enter the choice:\n");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter the number of data elements:\n");
                scanf("%d", &n);
                p = create(p, n);
                break;
            case 2:
                Preorder(p);
                printf("\n");
                break;
            case 3:
                Inorder(p);
                printf("\n");
                break;
            case 4:
                Postorder(p);
                printf("\n");
                break;
            case 5:
                printf("Enter the key to search:\n");
                scanf("%d", &key);
                flag = search(p, key);
                if (flag == 1) {
                    printf("key found\n");
                } else {
                    printf("key not found\n");
                }
                break;
            case 6:
                printf("EXIT POINT");
                done = 1;
                break;
            default:
                printf("Invalid choice\n");
        }
    }
    return 0;
}

NODE create(NODE root, int n) {
    int i, e;
    NODE q;
    if (root == NULL) {
        for (i = 1; i <= n; i++) {
            printf("enter data element\n");
            scanf("%d", &e);
            root = createtree(root, e);
        }
        return root;
    } else {
        printf("tree has already created\n");
        return root;
    }
}

NODE createtree(NODE p, int e) {
    if (p == NULL) {
        MALLOC(p, sizeof(struct tree), NODE);
        p->info = e;
        p->left = p->right = NULL;
        return p;
    } else if (e == p->info) {
        printf("duplicate key\n");
        return p;
    } else if (e < p->info) {
        p->left = createtree(p->left, e);
    } else {
        p->right = createtree(p->right, e);
    }
    return p;
```

```
}

int search(NODE p, int e) {
    if (p == NULL) {
        return 0;
    } else if (e == p->info) {
        return 1;
    } else if (e < p->info) {
        return search(p->left, e);
    } else {
        return search(p->right, e);
    }
}

void Preorder(NODE p) {
    if (p != NULL) {
        printf("%d\t", p->info);
        Preorder(p->left);
        Preorder(p->right);
    }
}

void Inorder(NODE p) {
    if (p != NULL) {
        Inorder(p->left);
        printf("%d\t", p->info);
        Inorder(p->right);
    }
```

```
}

void Postorder(NODE p) {
   if (p != NULL) {
       Postorder(p->left);
       Postorder(p->right);
       printf("%d\t", p->info);
   }
}
```

Output

```
1.Create        2.Preorder      3.Inorder       4.Postorder     5.Search        6.Exit
Enter the choice:
1
Enter the number of data elements:
2
enter data element
10
enter data element
20
1.Create        2.Preorder      3.Inorder       4.Postorder     5.Search        6.Exit
Enter the choice:
1
Enter the number of data elements:
2
tree has already created
1.Create        2.Preorder      3.Inorder       4.Postorder     5.Search        6.Exit
Enter the choice:
2
```

```
10        20

1.Create        2.Preorder      3.Inorder       4.Postorder     5.Search        6.Exit

Enter the choice:

3

10        20

1.Create        2.Preorder      3.Inorder       4.Postorder     5.Search        6.Exit

Enter the choice:

4

20        10

1.Create        2.Preorder      3.Inorder       4.Postorder     5.Search        6.Exit

Enter the choice:

5

Enter the key to search:

10

key found

1.Create        2.Preorder      3.Inorder       4.Postorder     5.Search        6.Exit

Enter the choice:

5

Enter the key to search:

100

key not found

1.Create2.Preorder      3.Inorder       4.Postorder     5.Search        6.Exit

Enter the choice:

6

EXIT POINT
```