10. Write a program to evaluate a postfix expression

```
#include<stdio.h>
#include<ctype.h>
int stack[20];
int top=-1;
void push(int x)
{
        stack[++top]=x;
}
int pop()
{
        return stack[top--];
}
void main()
{
        char exp[20];
        char *e;
        int n1,n2,n3,num;
        printf("Enter the expression:");
        scanf("%s",exp);
        e=exp;
        while(*e!='\0')
        {
                if(isdigit(*e))
                {
                        num=*e-(48);
                        push(num);
                }
                else
                {
                        n2=pop();
                        n1=pop();
                        switch(*e)
                        {
                                case '+': n3=n1+n2;
                                        break;
                                case '-': n3=n1-n2;
                                        break;
                                case '*': n3=n1*n2;
                                        break;
                                case '/': n3=n1/n2;
                                        break;
                        }
                        push(n3);
                }
                e++;
        }
        printf("\n the result of expression %s = %d",exp,pop());
}
```

Output:

```
Enter the expression:32+5/
the result of expression 32+5/ = 1
```

11. Write a program to insert the elements into a singly linked list and delete elements from the list. Display your list after each insertion and deletion.

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

struct node {
   int info;
   struct node *link;
}
*start, *nn, *curr, *temp, *prev;

void sll_insert();
void sll_delete();
void sll_display();

void main()
{
   int ch;
   start=NULL;

   while(1) {
      printf("\n operations on singly linked list");
      printf("\n1.Insert \t2.Delete \t3.Display \t4.Exit");
      printf("\n enter your choice:");
      scanf("%d",&ch);

      switch(ch) {
      case 1:
         sll_insert();
         break;
      case 2:
         sll_delete();
         break;
      case 3:
         sll_display();
         break;
      case 4:
         exit(0);
      default:
         printf("\n Invalid choice...");
      }
   }
}
```

```
void sll_insert()
{
    int i;
    nn=(struct node*)malloc(sizeof(struct node));
    if(nn==NULL) {
        printf("\n memory allocation failed...");
        return;
    }
    printf("\n enter the item:");
    scanf("%d",&i);
    nn->info=i;
    nn->link=NULL;
    curr=start;
    if(start==NULL)
        start=nn;
    else {
        while(curr->link != NULL) {
            curr = curr->link;
        }
        curr->link=nn;
    }
}

void sll_delete()
{
    int i,flag=0;
    printf("\n enter item to be deleted:");
    scanf("%d",&i);
    if(start->info == i) {
        flag = 1;
        start = start->link;
    } else {
        temp = start;
        prev = NULL;
        while(temp!=NULL) {
            if(temp->info==i) {
                flag=1;
                prev->link = temp->link;
                printf("\n node with information %d is deleted",i);
                free(temp);
                break;
            } else {
                prev = temp;
                temp=temp->link;
            }
        }
    }
    if(flag==0)
        printf("\n element not found...");
    else
        sll_display();
}

void sll_display()
{
    if(start==NULL) {
        printf("\n linked list is empty...");
        return;
    }
    temp=start;
    printf("\n linked list elements:");
    while(temp!=NULL) {
        printf("\t %d",temp->info);
        temp=temp->link;
    }
}
```

Output:

```
operations on singly linked list
1.Insert        2.Delete        3.Display       4.Exit
 enter your choice:1
 enter the item:10

 operations on singly linked list
1.Insert        2.Delete        3.Display       4.Exit
 enter your choice:1
 enter the item:20

 operations on singly linked list
1.Insert        2.Delete        3.Display       4.Exit
 enter your choice:1
 enter the item:30

 operations on singly linked list
1.Insert        2.Delete        3.Display       4.Exit
 enter your choice:1
 enter the item:40

 operations on singly linked list
1.Insert        2.Delete        3.Display       4.Exit
```

```
 enter your choice:3
 linked list elements:    10        20        30        40

 operations on singly linked list
1.Insert            2.Delete           3.Display         4.Exit
 enter your choice:2
 enter item to be deleted:20
 node with information 20 is deleted
 linked list elements:    10        30        40

  operations on singly linked list
1.Insert            2.Delete           3.Display         4.Exit
 enter your choice:4
```

12. Write a program to create a binary tree and find the height of the tree

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
        int value;
        struct node *left, *right;
};

struct node *create()
{
        struct node *p;
        int x;
        printf("Enter data(-1 for no data):");
                scanf("%d",&x);
        if(x==-1)
                return NULL;

        p=(struct node*)malloc(sizeof(struct node));
        p->value=x;

        printf("Enter left child of %d:\n",x);
        p->left=create();

        printf("Enter right child of %d:\n",x);
        p->right=create();

        return p;
}

void print(struct node *root)
{
         if (root!=NULL)
         {
                print(root->left);
                printf("%d \n", root->value);
                print(root->right);
         }
}
int max_h(int a, int b)
{
        return (a>b) ? a : b ;
}
int height(struct node* temp)
{
        if (temp == NULL)
                return -1;
        return max_h(height(temp->left), height(temp->right)) + 1;
}
void main()
{
        struct node *root=NULL, *s;
        int x;

        root=create();

        printf("The elements of the tree traversed in inorder way:\n");
        print(root);

        printf("Height of the tree : %d ", height(root));

}
```

Output:

```
Enter data(-1 for no data):10
Enter left child of 10:
Enter data(-1 for no data):20
Enter left child of 20:
Enter data(-1 for no data):30
Enter left child of 30:
Enter data(-1 for no data):-1
Enter right child of 30:
Enter data(-1 for no data):-1
Enter right child of 20:
Enter data(-1 for no data):-1
Enter right child of 10:
Enter data(-1 for no data):40
Enter left child of 40:
Enter data(-1 for no data):-1
Enter right child of 40:
Enter data(-1 for no data):-1
The elements of the tree traversed in inorder way:
30
20
10
40
Height of the tree : 2
```

13. **Write a program to create a binary search tree with the elements and perform inorder, preorder and post order traversal.**

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

struct Node
{
   int info;
   struct Node *left, *right;
} *temp, *root, *curr, *prev;

void createBST()
{
        int ele;
        printf("\nEnter the element: ");
        scanf("%d",&ele);

        temp=(struct Node*)malloc(sizeof(struct Node));
        temp->info = ele;
        temp->left = temp->right = NULL;

        if(root == NULL)
        {
                root = temp;
                return;
        }
        curr = root;
        while(curr!=NULL)
        {
                prev = curr;
                if(ele > curr->info)
                        curr = curr->right;
                else
                        curr = curr->left;
        }

        if(ele > prev->info)
                prev->right = temp;
        else
                prev->left = temp;
}

void postorder(struct Node* node)
{
   if (node==NULL)
```

```
         return;

   postorder(node->left);
   postorder(node->right);
   printf("%d ",node->info);
}

void inorder(struct Node* node)
{
   if(node==NULL)
        return;

   inorder(node->left);
   printf("%d ", node->info);
   inorder(node->right);
}

void preorder(struct Node* node)
{
   if(node==NULL)
        return;

   printf("%d ", node->info);
   preorder(node->left);
   preorder(node->right);
}

void main()
{
        int ch;
        root = NULL;

        while(1)
        {
                printf("\n 1. Create Binary Tree  2. Preorder  3. Inorder  4. Postorder\n");
                printf("Enter your choice\n");
                scanf("%d", &ch);

                switch(ch)
                {
                        case 1: createBST();
                                break;
                        case 2: printf("\nPreorder traversal is:\n");
                                preorder(root);
                                break;
                        case 3: printf("\nInorder traversal of is:\n");
```

```
                    inorder(root);
                    break;
            case 4: printf("\nPostorder traversal is:\n");
                    postorder(root);
                    break;
            default: exit(0);
        }
    }
}
```

Output:

```
1. Create Binary Tree  2. Preorder  3. Inorder  4. Postorder
Enter your choice
1
Enter the element: 10

1. Create Binary Tree  2. Preorder  3. Inorder  4. Postorder
Enter your choice
1
Enter the element: 20

1. Create Binary Tree  2. Preorder  3. Inorder  4. Postorder
Enter your choice
1
Enter the element: 5

1. Create Binary Tree  2. Preorder  3. Inorder  4. Postorder
Enter your choice
1
Enter the element: 30

1. Create Binary Tree  2. Preorder  3. Inorder  4. Postorder
Enter your choice
1
Enter the element: 12

1. Create Binary Tree  2. Preorder  3. Inorder  4. Postorder
Enter your choice
2
Preorder traversal is:
10 5 20 12 30

 1. Create Binary Tree  2. Preorder  3. Inorder  4. Postorder
Enter your choice
```

```
3
Inorder traversal of is:
5 10 12 20 30

 1. Create Binary Tree  2. Preorder  3. Inorder  4. Postorder
Enter your choice
4
Postorder traversal is:
5 12 30 20 10

 1. Create Binary Tree  2. Preorder  3. Inorder  4. Postorder
Enter your choice
5
```

14. Write a program to Sort the following elements using heap sort

```
#include<stdio.h>

void heapify(int*,int, int);
void heapsort(int*, int);
void print_array(int*, int);

void main()
{
   int i,arr[100],n;
   printf("Enter array size:");
   scanf("%d", &n);
   printf("\nEnter array elements: ");
   for(i=0; i<n; i++)
      scanf("%d", &arr[i]);
   printf("\nArray before sorting:\n");
   print_array(arr, n);

   heapsort(arr, n);

   printf("\n\nArray after sorting:\n");
   print_array(arr, n);
}

void heapsort(int* arr, int n)
{
   int i;
   for(i=n/2-1; i>=0; i--) {
      heapify(arr, n, i);
   }
   for(i=n-1; i>=0; i--) {
      int temp=arr[i];
      arr[i]=arr[0];
      arr[0]=temp;
      heapify(arr, i, 0);
   }
}

void heapify(int* arr, int n, int i)
{
   int largest=i;
   int left=2*i+1;
   int right=2*i+2;
   if(left<n&&arr[left]>arr[largest]) {
      largest=left;
```

```
    }
    if(right<n&&arr[right]>arr[largest]) {
        largest=right;
    }
    if(largest!=i) {
        int temp=arr[i];
        arr[i]=arr[largest];
        arr[largest]=temp;
        heapify(arr, n, largest);
    }
}

void print_array(int* arr, int n)
{
    int i;
    for(i=0; i<n; i++) {
        printf("%d ",arr[i]);
    }
}
```

Output:

```
Enter array size:6
Enter array elements:
29   3   43   12   56   67

Array before sorting:
29 3 43 12 56 67

Array after sorting:
3 12 29 43 56 67
```

15. Given two strings perform the following:

I. Find the length of the string

II. Concatenate the strings

III. Extract the substring

IV. Replace a string with another string

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<ctype.h>
void stringlength()
{
   char s[100];
   printf("Input of the string:\n");
   scanf("%s",s);
   printf("length of the string %s is %d \n",s,strlen(s));
}
void concatenate()
{
   char destination[100],source[100];
   printf("input first string:\n");
   scanf("%s",destination);
   printf("input second string: \n");
   scanf("%s",source);
   strcat(destination,source);
   printf("concatenated string:%s \n",destination);
}
void substring()
{
   char string[100],sub[100];
   int position,length,c=0;
   printf("input a string: \n");
   scanf("%s",string);
   printf("enter the position and length of substring: \n");
   scanf("%d%d",&position,&length);
   while(c<length) {
      sub[c]=string[position+c-1];
      c++;
   }
   sub[c]='\0';
   printf("required substring is \'%s\'\n",sub);
}
void characterreplace()
{
   char string[100],ch,replacech;
   int i;
```

```
    printf("input string:\n");
    scanf("%s",string);
    printf("input character to find:\n ");
    ch=getche();
    printf("input character to replace:");
    replacech=getche();
    for(i=0; i<strlen(string); i++) {
        if(string[i]==ch)
            string[i]=replacech;
    }
    printf("\nstring after replacing:\n");
    printf("%s\n",string);
}
int main()
{
    int choice;

    while(1) {
            printf("1. string length \t 2.string concatenation \n  3. substring extraction \t 4. character replacement \n");
        printf("enter your choice:");
        scanf("%d",&choice);
        switch(choice) {
        case 1:
            stringlength();
            break;
        case 2:
            concatenate();
            break;
        case 3:
            substring();
            break;
        case 4:
            characterreplace();
            break;
        default:
            exit(0);
        }
    }
}
```

Output:

```
1. string length       2.string concatenation
3. substring extracting   4. character replacement
enter your choice:1
Input of the string:
Hello
length of the string Hello is 5

1. string length       2.string concatenation
3. substring extracting   4. character replacement
enter your choice:2
input first string:
Hello
input second string:
 World
concatenated string:HelloWorld

1. string length       2.string concatenation
3. substring extracting   4. character replacement
enter your choice:3
input a string:
Indian
enter the position and length of substring:
5 2
required substring is 'an'

1. string length       2.string concatenation
3. substring extracting   4. character replacement
enter your choice:4
input string:
Hello
input character to find:
 l
input character to replace:
p
string after replacing:
Heppo
```