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

#define MAX 20

int top = -1;
char stack[MAX];

char push(char c)
{
 if(top == (MAX-1))
  printf("Stack Overflow\n");
 else
  stack[++top] =c;
}
char pop()
{
 if(top == -1)
  printf("Stack Underflow\n");
 else
  return stack[top--];
}
main()
{
 char str[20];
 int i;
 printf("Enter the string : " );
 gets(str);
 for(i=0;i<strlen(str);i++)
  push(str[i]);
 for(i=0;i<strlen(str);i++)
  str[i]=pop();
 printf("Reversed string is : ");
 puts(str);
}
```

Output1:

```
Enter the string : helloworld
Reversed string is : dlrowolleh
```

Problem 2:

```
#include<stdio.h>
#include<stack>
#include<string.h>
using namespace std;
stack<char>s;
bool isoperator(char c)
{
 if(c=='+'||c=='-'||c=='*'||c=='/')
  {
  return true;
 }
 else
 {
  return false;
 }
}
int order(char a)
{
 if(a=='*'||a=='/')
 {
  return 2;
 }
 if(a=='+'||a=='-')
 {
  return 1;
 }
}
bool highorder(char a,char b)
{
 if(order(a)<=order(b))
 {
  return 1;
 }
 else
 {
  return 0;
 }
}
bool isnumeric(char c)
{
 if(c>='0' && c<='9')
 {
  return 1;
 }
 else
 {
  return 0;
 }
}
bool isopening(char c)
{
 if(c=='(')
 {
  return 1;
 }
 else
 {
  return 0;
 }
}
bool isclosing(char c)
{
 if(c==')')
 {
  return 1;
 }
 else
 {
  return 0;
 }
}
char *infixtopostfix(char exp[])
{
 int i;
 char res[20]={'\0'};
 for(i=0;i<strlen(exp);i++)
 {
  if(isoperator(exp[i]))
  {
   while(!s.empty() && highorder(exp[i],s.top()) && !isopening(s.top()))
   {
    strncat(res,&s.top(),1);
    s.pop();
   }
   s.push(exp[i]);
  }
  else if(isnumeric(exp[i]))
  {
   strncat(res,&exp[i],1);
  }
  else if(isopening(exp[i]))
  {
   s.push(exp[i]);
  }
  else if(isclosing(exp[i]))
  {
   while(!s.empty() && !isopening(s.top()))
   {
    strncat(res,&s.top(),1);
    s.pop();
   }
   s.pop();
  }
 }
 while(!s.empty())
 {
  strncat(res,&s.top(),1);
  s.pop();
 }
 return res;
}
main()
{
 char exp[20],r[20];
 printf("enter the expression");
 scanf("%s",exp);
 strcpy(r,infixtopostfix(exp));
 printf("%s",r);
}
```
Out put:
```
enter the expression1+3*5-1+2/2+6/3
135*+1-22/+63/+
--------------------------------
process exited aftern94.13vsecounds withreturn value 0
press any key to continue...
```

Problem 3:

```
#include<stdio.h>
#include<stack>
using namespace std;
stack<int>s1;
stack<int>s2;
main()
{
 int i,n,e;
 printf("enter the no of elements");
 scanf("%d",&n);
 for(i=0;i<n;i++)
 {
  printf("enter the element");
  scanf("%d",&e);
  s1.push(e);
 }
 printf("removing the bottom element:\n");
 while(!s1.empty())
 {
  s2.push(s1.top());
  s1.pop();
 }
 s2.pop();
 while(!s2.empty())
 {
  s1.push(s2.top());
  s2.pop();
 }
 printf("after removing the front element:\n");
 while(!s1.empty())
 {
  printf("%d",s1.top());
  s1.pop();
 }

}
```
Output:
```
 enter the no of elements5
```

```
enter the element1
enter the element2
enter the element3
enter the element4
enter the element5
removing the bottom element:
after removing the front element:
5432
--------------------------------
process exited after 6.793 secounds with return value 0
press any key to continue...
```

Problem 4:

```
#include<stdio.h>
#include<stdlib.h>
struct node{
 int data;
 struct node *leftlink;
 struct node *rightlink;
}*root=NULL;
struct node* insert(struct node* root,int e)
{
 if(root==NULL)
 {
  root=(struct node*)malloc(sizeof(struct node));
  root->data=e;
  root->leftlink=root->rightlink=NULL;
  return root;
 }
 else if(root->data>e)
 {
  root->leftlink=insert(root->leftlink,e);
 }
 else if(root->data<e)
 {
  root->rightlink=insert(root->rightlink,e);
 }
 return root;
}
int minimum(struct node* root)
{
 if(root->leftlink==NULL)
 {
  return root->data;
 }
 else
 {
  return minimum(root->leftlink);
 }
}
```

```
struct node* remove(struct node* root,int e)
{
 if(root==NULL)
 {
  return root;
 }
 else if(root->data>e)
 {
  root->leftlink=remove(root->leftlink,e);
 }
 else if(root->data<e)
 {
  root->rightlink=remove(root->rightlink,e);
 }
 else
 {
  if(root->leftlink==NULL && root->rightlink==NULL)


   {
   delete root;
   return NULL;
  }
  else if(root->leftlink==NULL)
  {
   root=root->rightlink;
  }
  else if(root->rightlink==NULL)
  {
   root=root->leftlink;
  }
  else
  {
   int key=minimum(root->rightlink);
   root->data=key;
   root->rightlink=remove(root->rightlink,key);
  }
 }
 return root;
}
void inorder(struct node *root)
{
 if(root==NULL)
 {
  return;
 }
 inorder(root->leftlink);
 printf("%d",root->data);
 inorder(root->rightlink);
}
main()
{
 int n,i,e;
 printf("enter no of elements");
 scanf("%d",&n);
 for(i=0;i<n;i++)
 {
  printf("enter the element");
  scanf("%d",&e);
  root=insert(root,e);
 }
 inorder(root);
 printf("\n");
 printf("enter the element to to remove");
 scanf("%d",&e);
 root=remove(root,e);
 inorder(root);
}
```

Out put:

```
enter no of elements9
enter the element8
enter the element3
enter the element10
enter the element1
enter the element6
enter the element14
enter the element4
enter the element7
enter the element13
134678101314
enter the element to remove3
14678101314
----------------------------
process exited after 26.44 secounds with return value 0
press any key to countinue...
```