Q18. WAP in C to implement shift reduce parser for the following grammar:

E → 2E2
E → 3E3
E → 4

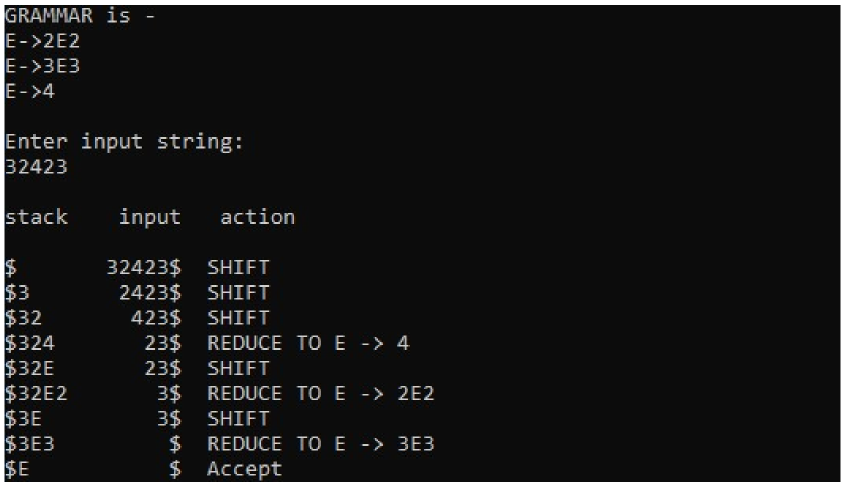
```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int k=0,z=0,i=0,j=0,c=0;
char a[16],ac[20],stk[15],act[10];

void check()
{
        strcpy(ac,"REDUCE TO E -> ");
        for(z=0; z<c; z++)
        if(stk[z]=='4')
    {
                printf("%s4",ac);
                stk[z]='E';
            stk[z+1]='\0';
        printf("\n$%s\t%s$\t",stk,a);
    }
        for(z=0; z<c; z++)
        if(stk[z]=='2' && stk[z+1]=='E' && stk[z+2]=='2')
    {
                printf("%s2E2",ac);
        stk[z]='E';
            stk[z+1]='\0';
        stk[z+2]='\0';
            printf("\n$%s\t%s$\t",stk,a);
        i=i-2;
    }
        for(z=0; z<c; z++)
        if(stk[z]=='3' && stk[z+1]=='E' && stk[z+2]=='3')
    {
                printf("%s3E3",ac);
                stk[z]='E';
            stk[z+1]='\0';
        stk[z+1]='\0';
            printf("\n$%s\t%s$\t",stk,a);
        i=i-2;
    }
}
int main()
{

    puts("GRAMMAR is -\nE->2E2 \nE->3E3 \nE->4\n");
    puts("Enter input string: ");
    scanf("%[^\n]",a);
    c=strlen(a);
    strcpy(act,"SHIFT");
    puts("\nstack \t input \t action");
    printf("\n$\t%s$\t",a);
    for(k=0,i=0; j<c; k++,i++,j++)
    {
            printf("%s",act);
            stk[i]=a[j];
            stk[i+1]='\0';
            a[j]=' ';
            printf("\n$%s\t%s$\t",stk,a);
            check();
    }
```

```
        check();
        if(stk[0]=='E'&& stk[1]=='\0')
                printf("Accept\n");
        else
                printf("Reject\n");
}
```

OUTPUT:

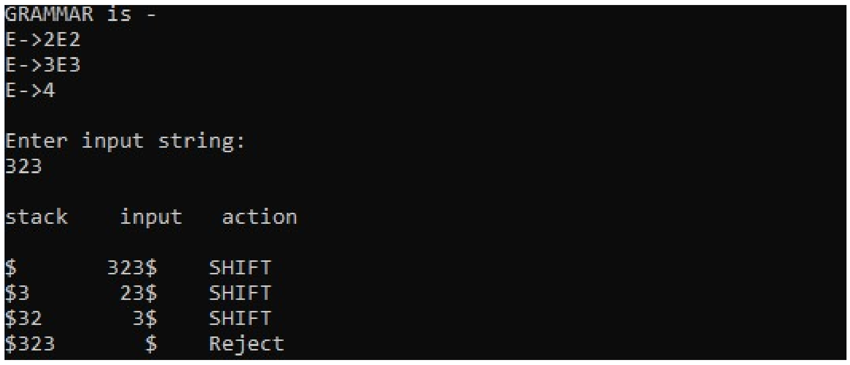


```
GRAMMAR is -
E->2E2
E->3E3
E->4

Enter input string:
32423

stack    input   action

$       32423$  SHIFT
$3       2423$  SHIFT
$32       423$  SHIFT
$324       23$  REDUCE TO E -> 4
$32E       23$  SHIFT
$32E2       3$  REDUCE TO E -> 2E2
$3E         3$  SHIFT
$3E3         $  REDUCE TO E -> 3E3
$E           $  Accept
```



```
GRAMMAR is -
E->2E2
E->3E3
E->4

Enter input string:
323

stack    input   action

$       323$    SHIFT
$3       23$    SHIFT
$32       3$    SHIFT
$323       $    Reject
```

Q19. WAP in C to implement operator precedence parser for the following grammar:

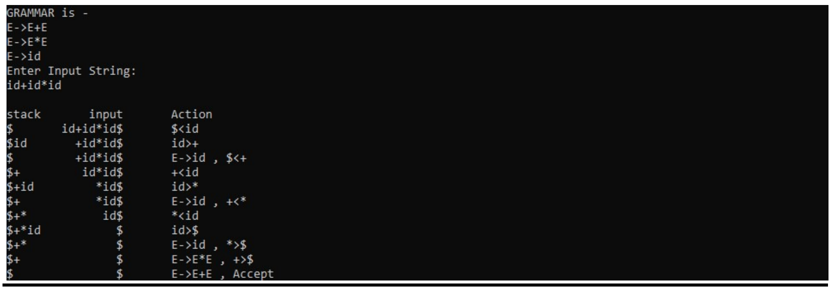
E → E + E
E → E * E
E → id

```
#include<bits/stdc++.h>
using namespace std;
stack<string> st,myst;
string input,temp="",top;
int i=0,var=0;

int check(string a,string b)
{
        if(a=="$" && b=="$")
                return 1;
        if(a=="id"&&(b=="+"||b=="*"||b=="$"))
                return 2;
        if(a=="*"&&(b=="+"||b=="$"))
                return 2;
        if(a=="+"&&b=="$")
                return 2;
        if(a==b && b=="id")
                return 4;
        return 3;
}
int main()
{
        cout<<"GRAMMAR is -\nE->E+E\nE->E*E\nE->id\n Enter Input String:\n ";
        cin>>input;
        input+="$";
        st.push("$");
        cout<<"\nstack \t    input \tAction\n";
        for(i=0;i<input.length();)
        {
                string s1="";
                while(!st.empty())
                {
                        top=st.top();
                        st.pop();
                        myst.push(top);
                }
                while(!myst.empty())
                {
                        top=myst.top();
                        s1+=top;
                        myst.pop();
                        st.push(top);
                }
                top=st.top();
                cout<<s1<<"\t"<<input<<"\t";
                s1="";
                if(input[i]=='i' && input[i+1]=='d')
                {
                        input[i]=' ';
                        input[i+1]=' ';
                        i+=2;
                        var=2;
                        s1="id";
                }
                else
                {
                        s1=input[i];
                        input[i]=' ';
                        var=1;
                        i++;
                }
                switch(check(s1,top))
                {
                        case 1:
                                cout<<temp<<"Accept"<<endl;
                                temp=""; break;
                        case 2:
                                st.push(s1);
                                cout<<temp<<top<<"<"<<s1<<endl;
                                temp="";break;
                        case 3:
                                cout<<temp<<top<<">"<<s1<<endl;
                                temp="";
                                if(top=="id")
                                        temp="E->id , ";
                                else if(top=="*")
                                        temp="E->E*E , ";
                                else if(top=="+")
                                        temp="E->E+E , ";
                                if(var==2)
                                {
                                        i-=var;
                                        input[i]='i';
```

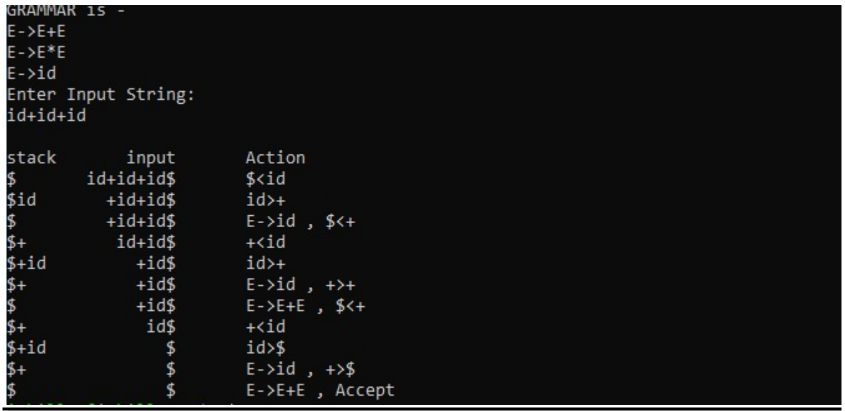
```
                                                input[i+1]='d';
                                        }
                                        if(var==1)
                                        {
                                                i-=var;
                                                input[i]=s1[0];
                                        }
                                        st.pop();break;
                                default:
                                        cout<<"\nError\n";return 0;
                        }
                }
}
```

OUTPUT:



```
GRAMMAR is -
E->E+E
E->E*E
E->id
Enter Input String:
id+id*id

stack       input       Action
$       id+id*id$       $<id
$id       +id*id$       id>+
$         +id*id$       E->id , $<+
$+         id*id$       +<id
$+id         *id$       id>*
$+           *id$       E->id , +<*
$+*           id$       *<id
$+*id           $       id>$
$+*             $       E->id , *>$
$+              $       E->E*E , +>$
$               $       E->E+E , Accept
```



```
GRAMMAR is -
E->E+E
E->E*E
E->id
Enter Input String:
id+id+id

stack       input       Action
$       id+id+id$       $<id
$id       +id+id$       id>+
$         +id+id$       E->id , $<+
$+         id+id$       +<id
$+id         +id$       id>+
$+           +id$       E->id , +>+
$            +id$       E->E+E , $<+
$+            id$       +<id
$+id            $       id>$
$+              $       E->id , +>$
$               $       E->E+E , Accept
```