```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
char prod[7][10]={"E->TX","T->(E)","T->iY","X->+E","X->@","Y->*T","Y->@"};
//char first[4][10]={"(i","(i","+@","*@"};
char first[7][10]={"(i","(i","(i","+","@@","*","@@@"};
//char follow[4][10]={"$)","$)+","$)","$)+"};
char follow[7][10]={"$)","$)+","$)+","$)","$)","$)+","$)+"};
char table[5][7][10];
char input[10];
int top=-1;
char stack[25];
char curp[20];
void push(char item)
{
stack[++top]=item;
}
void pop()
{
top--;
}
void display()
{
int i;
for(i=top;i>=0;i--)
printf("%c",stack[i]);
}
int numr(char c)
{
switch(c)
{
case 'E':return 1;
case 'T':return 2;
case 'X':return 3;
case 'Y':return 4;
case '(':return 1;
case ')':return 2;
case '+':return 3;
case '*':return 4;
case 'i':return 5; //int
case '$':return 6;
}
return(1);
```

```
}
void main()
{
char c;
int i,j,k,n;
for(i=0;i<5;i++)
for(j=0;j<7;j++)
strcpy(table[i][j],"e");

printf("\nGrammar:\n");
for(i=0;i<7;i++)
printf("%s\n",prod[i]);
printf("\nfirst={%s%s%s%s%s%s%s}",first[0],first[1],first[2],first[3],first[4],first[5],first[6]);
printf("\nfollow={%s%s%s%s%s%s%s}\n",follow[0],follow[1],follow[2],follow[3],follow[4],follow[5],
follow[6]);
printf("\nPredictive parsing table for the given grammar\n");
strcpy(table[0][0]," ");
strcpy(table[0][1],"(");
strcpy(table[0][2],")");
strcpy(table[0][3],"+");
strcpy(table[0][4],"*");
strcpy(table[0][5],"i");
strcpy(table[0][6],"$");
strcpy(table[1][0],"E");
strcpy(table[2][0],"T");
strcpy(table[3][0],"X");
strcpy(table[4][0],"Y");
for(i=0;i<7;i++)
{
k=strlen(first[i]);
for(j=0;j<k;j++)
if(first[i][j]!='@')
strcpy(table[numr(prod[i][0])][numr(first[i][j])],prod[i]);
else
strcpy(table[numr(prod[i][0])][numr(follow[i][j])],prod[i]);

}
printf("\n----------------------------------------------------------\n");
for(i=0;i<5;i++)
for(j=0;j<7;j++)
{
printf("%-10s",table[i][j]);
if(j==6)
```

```
printf("\n----------------------------------------------------------------\n");
}
printf("Enter the input string terminated with $ to parse:-");
scanf("%s",&input);
for(i=0;input[i]!='\0';i++){
if((input[i]!='i')&&(input[i]!='+')&&(input[i]!='*')&&(input[i]!='(')&&(input[i]!=')')&&(input[i]!='$'))
{
printf("Invalid String");
exit(0);
}}
if(input[i-1]!='$')
{
printf("\n\nInput String entered without end marker $");
exit(0);
}
push('$');
push('E');
i=0;
printf("\n\n");
printf("Stack\t\tInput\t\tAction");
printf("\n----------------------------------------------------------------\n");
while(input[i]!='$' || stack[top]!='$')
{
display();
printf("\t\t%s\t",(input+i));
if(stack[top]==input[i])
{
printf("\tmatched%c\n",input[i]);
pop();
i++;
}
else
{
if(stack[top]>=65 && stack[top]<=92)
{
strcpy(curp,table[numr(stack[top])][numr(input[i])]);
if(!strcmp(curp,"e"))
{

printf("\nInvalid String Rejected");
exit(0);
}
else
```

```
{
printf("\tApply Production%s\n",curp);
}
if(curp[3]=='@')
pop();
else
{
pop();
n=strlen(curp);
for(j=n-1;j>=3;j--){
push(curp[j]);}
}
}
}
}

display();
printf("\t\t%s\t",(input+i));
//printf("\n----------------------------------------------------\n");
if(stack[top]=='$' && input[i]=='$')
{
printf("\tValid String-Accepted\n");
}
else
{
printf("\tInvalid String-Rejected\n");
}
}
```