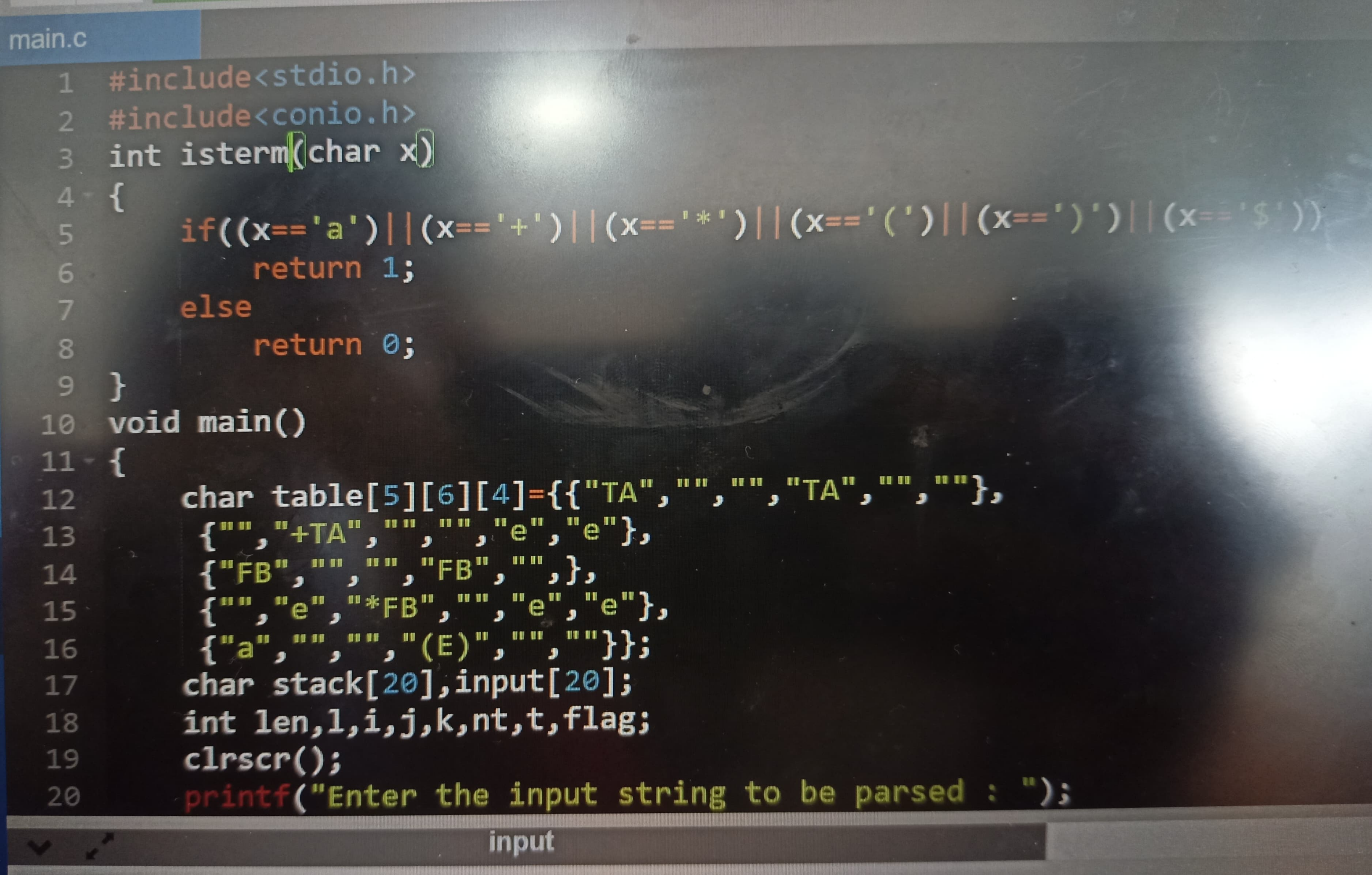
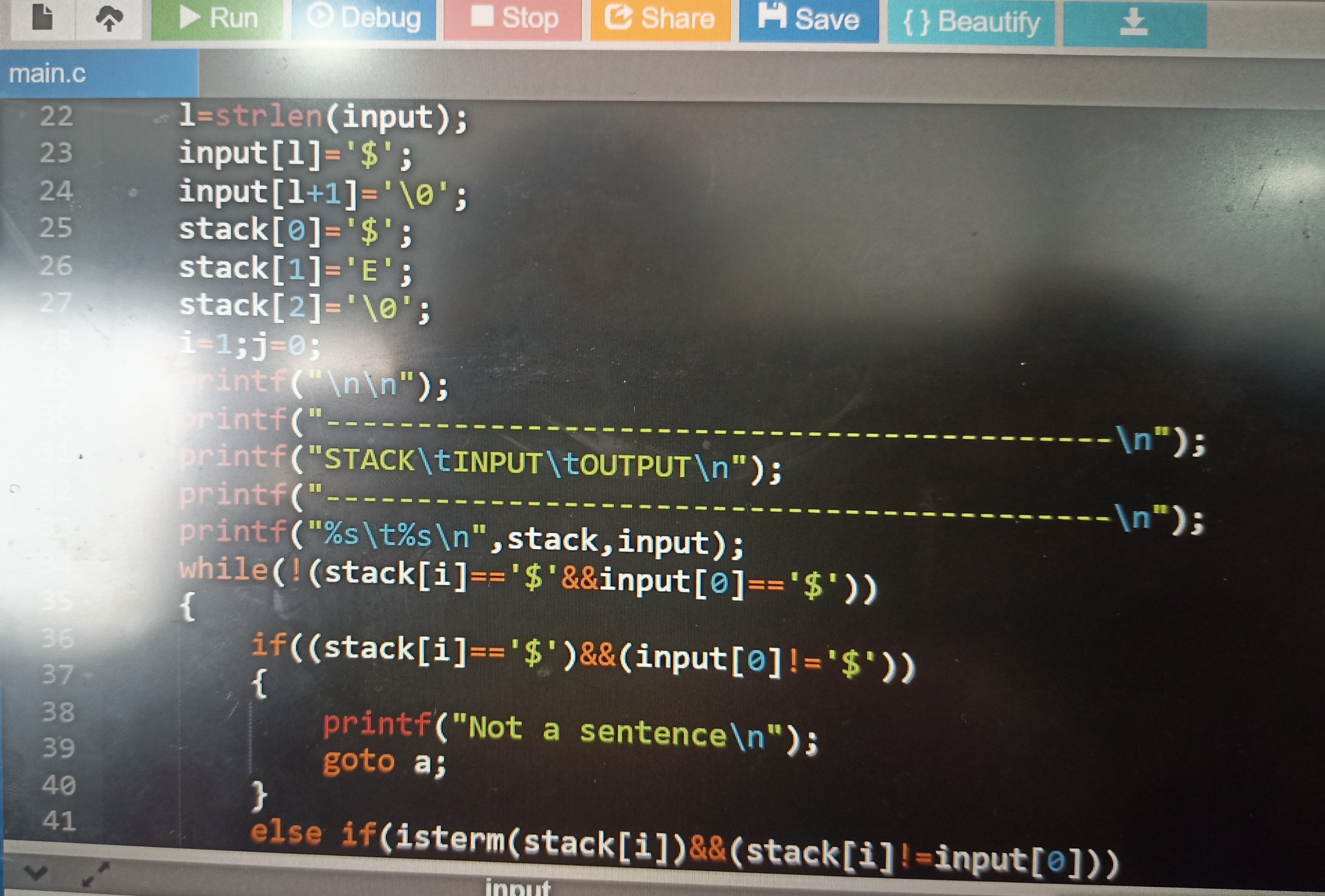
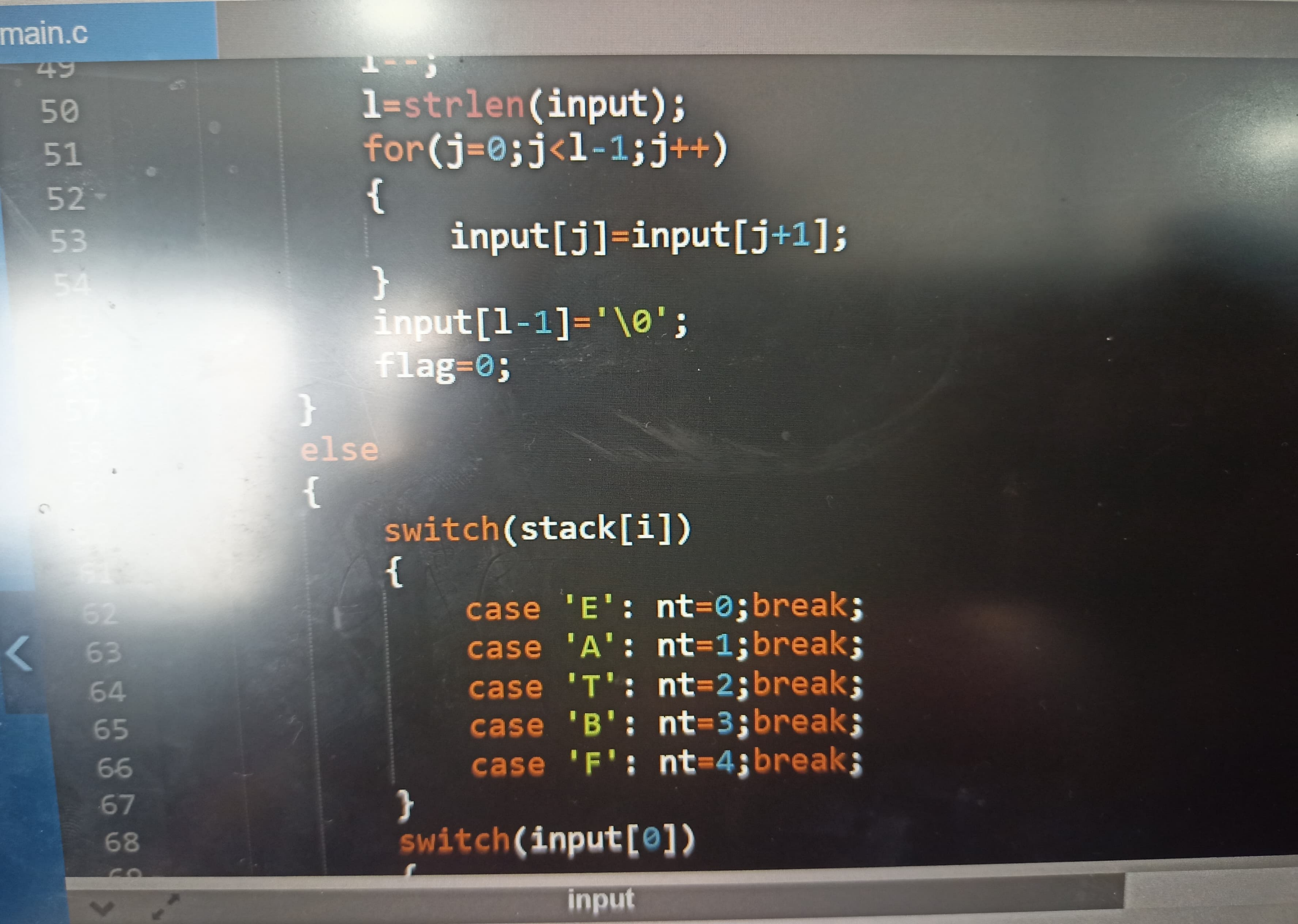


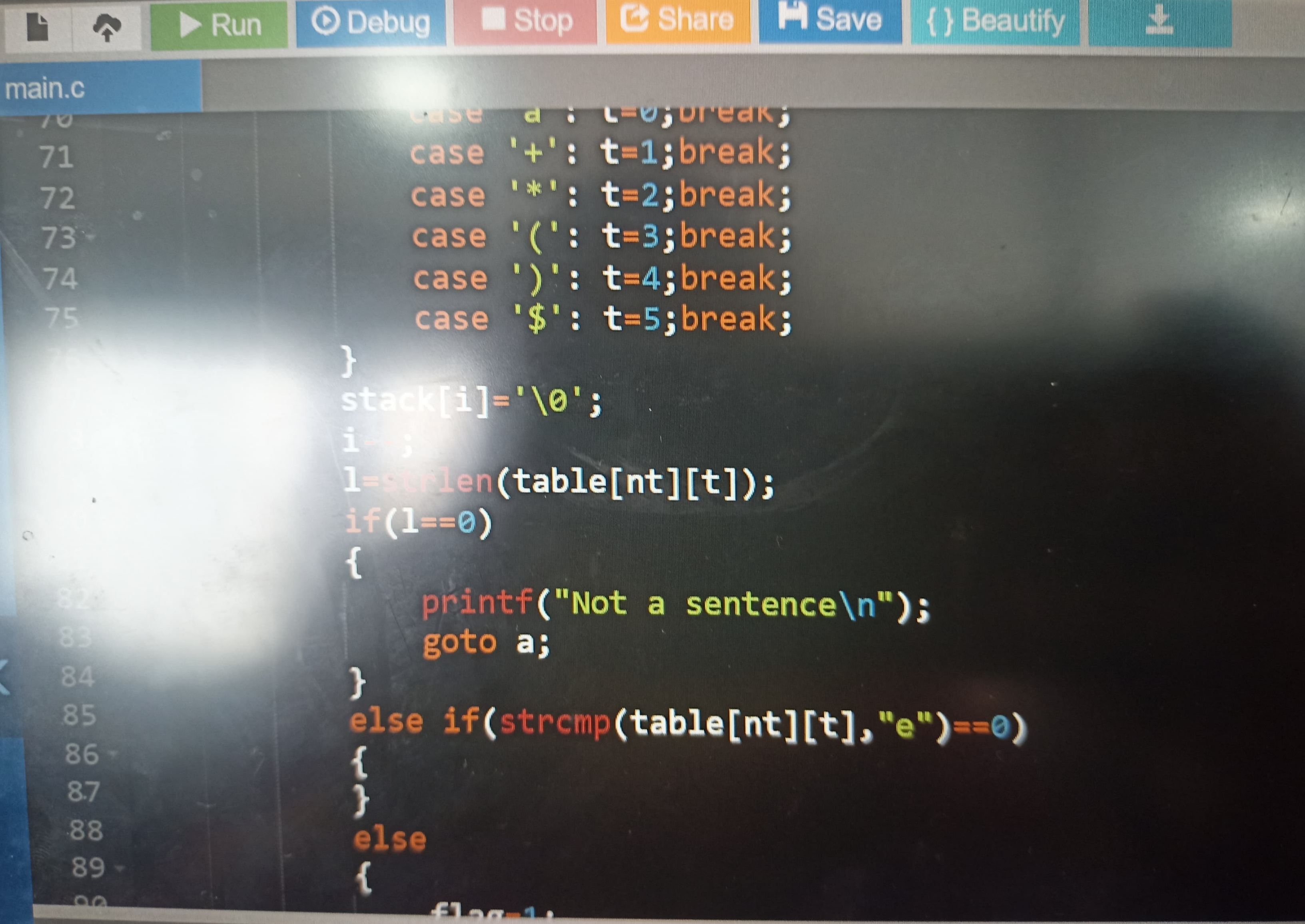
```c
#include<stdio.h>
#include<conio.h>
int isterm(char x)
{
    if((x=='a')||(x=='+')||(x=='*')||(x=='(')||(x==')')||(x=='$'))
        return 1;
    else
        return 0;
}
void main()
{
    char table[5][6][4]={{"TA","","","TA","",""},
     {"","+TA","","","e","e"},
     {"FB","","","FB","",},
     {"","e","*FB","","e","e"},
     {"a","","","(E)","",""}};
    char stack[20],input[20];
    int len,l,i,j,k,nt,t,flag;
    clrscr();
    printf("Enter the input string to be parsed : ");
```



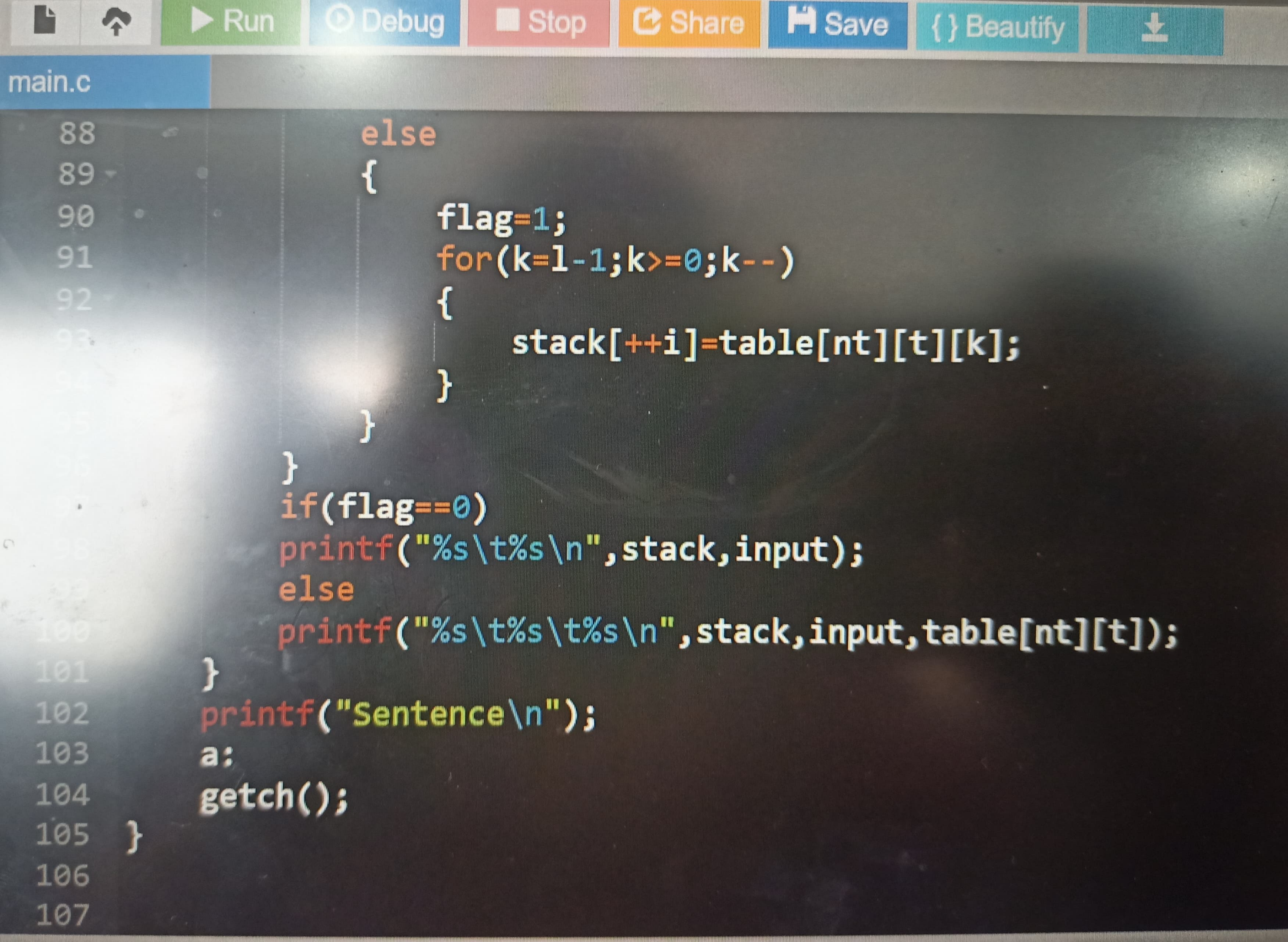
```
    l=strlen(input);
    input[l]='$';
    input[l+1]='\0';
    stack[0]='$';
    stack[1]='E';
    stack[2]='\0';
    i=1;j=0;
    printf("\n\n");
    printf("--------------------------------------------\n");
    printf("STACK\tINPUT\tOUTPUT\n");
    printf("--------------------------------------------\n");
    printf("%s\t%s\n",stack,input);
    while(!(stack[i]=='$'&&input[0]=='$'))
    {
        if((stack[i]=='$')&&(input[0]!='$'))
        {
            printf("Not a sentence\n");
            goto a;
        }
        else if(isterm(stack[i])&&(stack[i]!=input[0]))
```



```c
        i--;
        l=strlen(input);
        for(j=0;j<l-1;j++)
        {
            input[j]=input[j+1];
        }
        input[l-1]='\0';
        flag=0;
    }
    else
    {
        switch(stack[i])
        {
            case 'E': nt=0;break;
            case 'A': nt=1;break;
            case 'T': nt=2;break;
            case 'B': nt=3;break;
            case 'F': nt=4;break;
        }
        switch(input[0])
        {
```

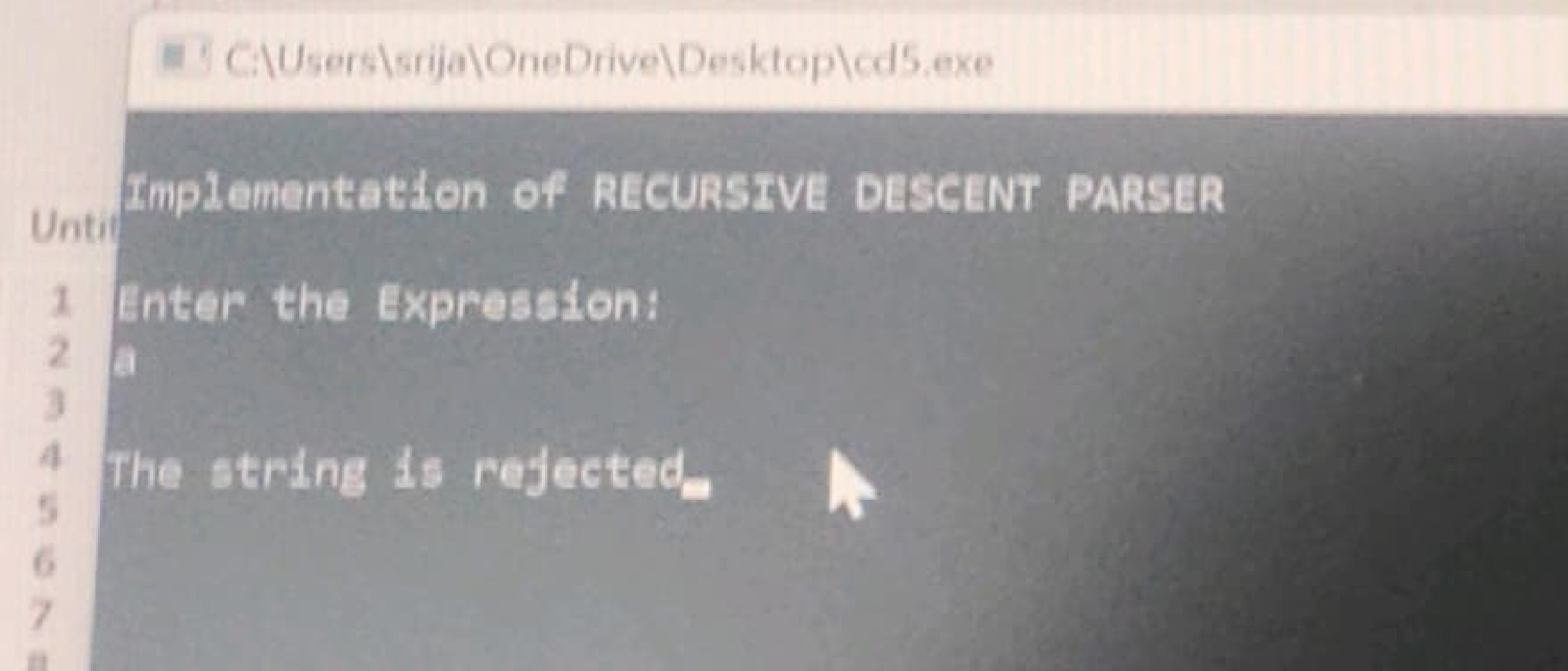


```
        case 'a': t=0;break;
        case '+': t=1;break;
        case '*': t=2;break;
        case '(': t=3;break;
        case ')': t=4;break;
        case '$': t=5;break;
    }
    stack[i]='\0';
    i--;
    l=strlen(table[nt][t]);
    if(l==0)
    {
        printf("Not a sentence\n");
        goto a;
    }
    else if(strcmp(table[nt][t],"e")==0)
    {
    }
    else
    {
        flag=1;
```



```c
            else
            {
                flag=1;
                for(k=l-1;k>=0;k--)
                {
                    stack[++i]=table[nt][t][k];
                }
            }
        }
        if(flag==0)
        printf("%s\t%s\n",stack,input);
        else
        printf("%s\t%s\t%s\n",stack,input,table[nt][t]);
    }
    printf("Sentence\n");
    a:
    getch();
}


```



```
Implementation of RECURSIVE DESCENT PARSER

Enter the Expression:
a

The string is rejected
```