

10. Write a C program to construct recursive descent parsing.

```
#include <stdio.h>
#include <string.h>
#define SUCCESS 1
#define FAILED 0
int E(), Edash(), T(), Tdash(), F();
const char *cursor;
char string[64];
int main()
{
    puts("Enter the string");
    // scanf("%s", string);
    sscanf("i+(i+i)*i", "%s", string);
    cursor = string;
    puts("");
    puts("Input    Action");
    puts("-----");
    if (E() && *cursor == '\0') {
        puts("-----");
        puts("String is successfully parsed");
        return 0;
    } else {
        puts("-----");
        puts("Error in parsing String");
        return 1;
    }
}
int E()
{
    printf("%%-16s E -> T E\n", cursor);
    if (T()) {
        if (Edash())
            return SUCCESS;
        else
            return FAILED;
    } else
        return FAILED;
}
int Edash()
{
    if (*cursor == '+') {
        printf("%%-16s E' -> + T E\n", cursor);
        cursor++;
        if (T())
        {
            if (Edash())
                return SUCCESS;
            else
                return FAILED;
        }
    }
}
```

```

    }
    else
        return FAILED;
}
else
{
    printf("%%-16s E' -> $\n", cursor);
    return SUCCESS;
}
}

```

```

int T()
{
    printf("%%-16s T -> F T'\n", cursor);
    if (F())
    {
        if (Tdash())
            return SUCCESS;
        else
            return FAILED;
    } else
        return FAILED;
}

```

```

int Tdash()
{
    if (*cursor == '*')
    {
        printf("%%-16s T' -> * F T'\n", cursor);
        cursor++;
        if (F())
        {
            if (Tdash())
                return SUCCESS;
            else
                return FAILED;
        } else
            return FAILED;
    }
    else
    {
        printf("%%-16s T' -> $\n", cursor);
        return SUCCESS;
    }
}

```

```

int F()
{
    if (*cursor == '(')
    {

```

```

printf("%-16s F -> ( E )\n", cursor);
cursor++;
if (E())
{
    if (*cursor == ')')
    {
        cursor++;
        return SUCCESS;
    } else
        return FAILED;
} else
    return FAILED;
} else if (*cursor == 'i')
{
    cursor++;
    printf("%-16s F ->i\n", cursor);
    return SUCCESS;
} else
    return FAILED;

```

The screenshot shows a C++ IDE with a project named 'exp 10.cpp'. The main window displays the output of a parser execution. The output is as follows:

```

Enter the string
Input      Action
-----
i+(i+i)*i  E -> T E'
i+(i+i)*i  T -> F T'
+(i+i)*i   F -> i
+(i+i)*i   T' -> $
+(i+i)*i   E' -> + T E'
(i+i)*i    T -> F T'
(i+i)*i    F -> ( E )
(i+i)*i    E -> T E'
(i+i)*i    T -> F T'
+i)*i      F -> i
+i)*i      T' -> $
+i)*i      E' -> + T E'
i)*i       T -> F T'
)*i        F -> i
)*i        T' -> $
)*i        E' -> $
*i         T' -> * F T'
*i         F -> i
*i         T' -> $
*i         E' -> $

String is successfully parsed

Process exited after 5.648 seconds with return value 0
Press any key to continue . . .

```

The status bar at the bottom of the IDE shows: Line: 10, Col: 33, Sel: 0, Lines: 117, Length: 2700, Insert, Done parsing in 0.031 seconds.