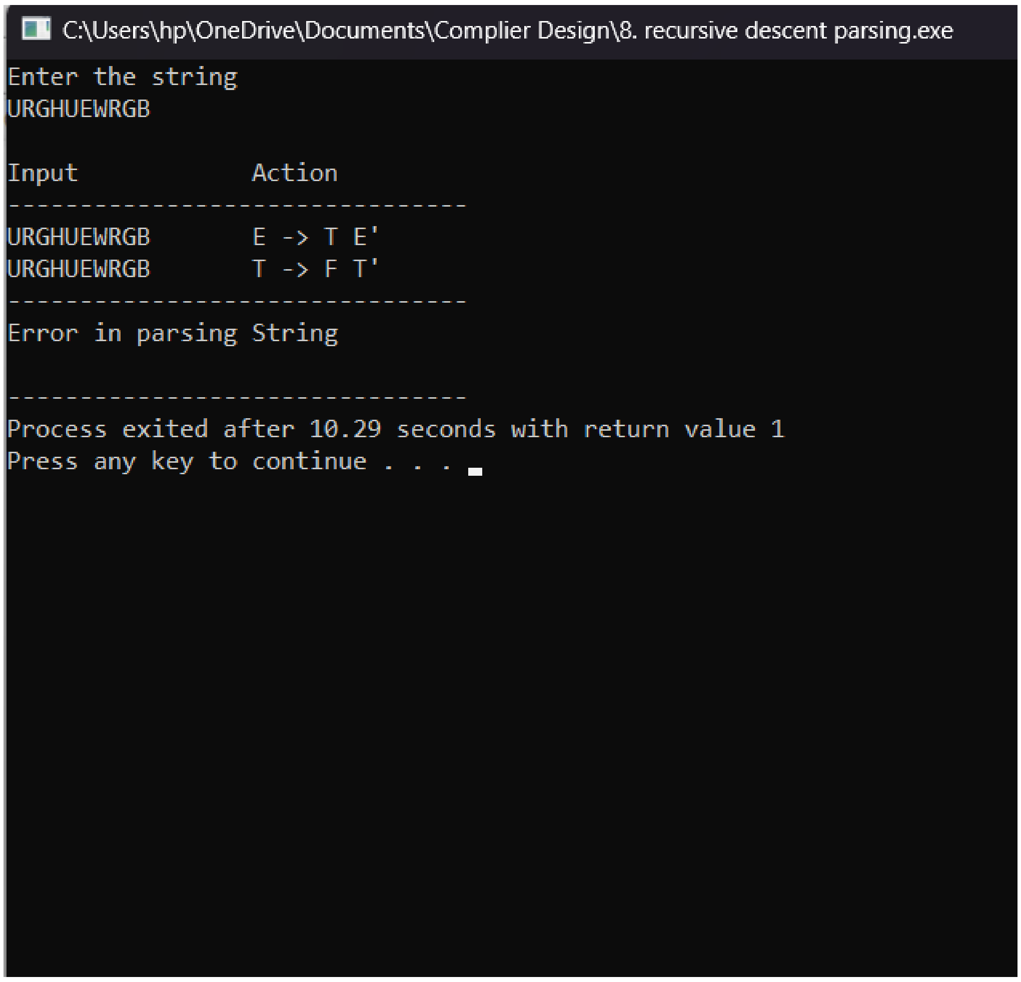
8. recursive descent parsing

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

#define SUCCESS 1
#define FAILED 0

// Function prototypes
int E(), Edash(), T(), Tdash(), F();

const char *cursor;
char string[64];

int main() {
        puts("Enter the string");
        scanf("%s", string); // Read input from the user
        cursor = string;
        puts("");
        puts("Input                 Action");
        puts("--------------------------------");

        // Call the starting non-terminal E
        if (E() && *cursor == '\0') { // If parsing is successful and the cursor has reached the
end
                puts("--------------------------------");
                puts("String is successfully parsed");
                return 0;
        }
        else {
                puts("--------------------------------");
                puts("Error in parsing String");
                return 1;
        }
}

// Grammar rule: E -> T E'
int E() {
        printf("%-16s E -> T E'\n", cursor);
        if (T()) { // Call non-terminal T
                if (Edash()) // Call non-terminal E'
                        return SUCCESS;
                else
                        return FAILED;
        }
```

```
	else
		return FAILED;
}

// Grammar rule: E' -> + T E' | $
int Edash() {
	if (*cursor == '+') {
		printf("%-16s E' -> + T E'\n", cursor);
		cursor++;
		if (T()) { // Call non-terminal T
			if (Edash()) // Call non-terminal E'
				return SUCCESS;
			else
				return FAILED;
		}
		else
			return FAILED;
	}
	else {
		printf("%-16s E' -> $\n", cursor);
		return SUCCESS;
	}
}

// Grammar rule: T -> F T'
int T() {
	printf("%-16s T -> F T'\n", cursor);
	if (F()) { // Call non-terminal F
		if (Tdash()) // Call non-terminal T'
			return SUCCESS;
		else
			return FAILED;
	}
	else
		return FAILED;
}

// Grammar rule: T' -> * F T' | $
int Tdash() {
	if (*cursor == '*') {
		printf("%-16s T' -> * F T'\n", cursor);
		cursor++;
		if (F()) { // Call non-terminal F
			if (Tdash()) // Call non-terminal T'
				return SUCCESS;
```

```
                else
                        return FAILED;
            }
            else
                return FAILED;
        }
        else {
            printf("%-16s T' -> $\n", cursor);
            return SUCCESS;
        }
    }

    // Grammar rule: F -> ( E ) | i
    int F() {
        if (*cursor == '(') {
            printf("%-16s F -> ( E )\n", cursor);
            cursor++;
            if (E()) { // Call non-terminal E
                if (*cursor == ')') {
                    cursor++;
                    return SUCCESS;
                }
                else
                    return FAILED;
            }
            else
                return FAILED;
        }
        else if (*cursor == 'i') {
            printf("%-16s F -> i\n", cursor);
            cursor++;
            return SUCCESS;
        }
        else
            return FAILED;
    }
```

OUTPUT:


C:\Users\hp\OneDrive\Documents\Complier Design\8. recursive descent parsing.exe

```
Enter the string
URGHUEWRGB

Input            Action
--------------------------------
URGHUEWRGB       E -> T E'
URGHUEWRGB       T -> F T'
--------------------------------
Error in parsing String

--------------------------------
Process exited after 10.29 seconds with return value 1
Press any key to continue . . .
```