Exp. No. 12

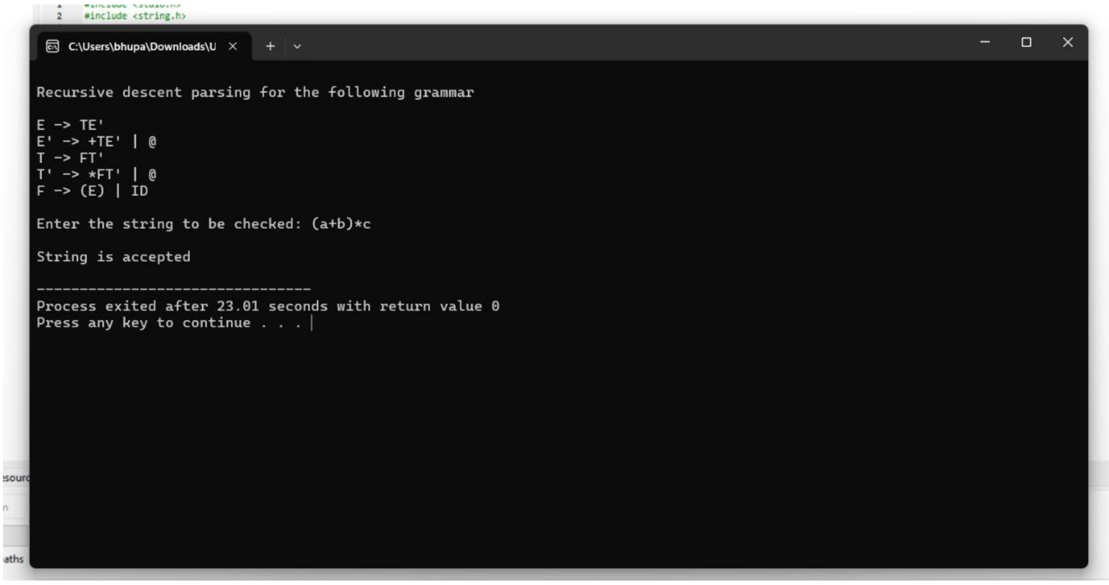
Write a C program to construct recursive descent parsing for the given grammar

E → TE’
E’ → +TE’ / ∈
T → FT’
T’ → *FT’ / ∈
F → (E) / id

PROGRAM

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

char input[100];
int i = 0, l;

int E();  // Declaration of functions
int EP();
int T();
int TP();
int F();

int main() {
   printf("\nRecursive descent parsing for the following grammar\n");
   printf("\nE -> TE'\nE' -> +TE' | @\nT -> FT'\nT' -> *FT' | @\nF -
> (E) | ID\n");
   printf("\nEnter the string to be checked: ");
   fgets(input, sizeof(input), stdin);

   // Remove newline character if present
   input[strcspn(input, "\n")] = '\0';

   l = strlen(input);

   if (E()) {
      if (input[i] == '\0') {
         printf("\nString is accepted\n");
      } else {
         printf("\nString is not accepted\n");
      }
   } else {
        printf("\nString not accepted\n");
    }

    return 0;
}

int E() {
    if (T()) {
        if (EP()) {
            return 1;
        } else {
            return 0;
        }
    } else {
        return 0;
    }
}

int EP() {
    if (input[i] == '+') {
        i++;
        if (T()) {
            if (EP()) {
                return 1;
            } else {
                return 0;
            }
        } else {
            return 0;
        }
    } else {
        return 1;
    }
}

int T() {
    if (F()) {
        if (TP()) {
            return 1;
        } else {
            return 0;
        }
    } else {
        return 0;
    }
}

int TP() {
    if (input[i] == '*') {
        i++;
        if (F()) {
            if (TP()) {
                return 1;
            } else {
                return 0;
            }
        } else {
            return 0;
        }
    } else {
        return 1;
    }
}

int F() {
    if (input[i] == '(') {
        i++;
        if (E()) {
            if (input[i] == ')') {
                i++;
                return 1;
            } else {
                return 0;
            }
        } else {
            return 0;
        }
    } else if ((input[i] >= 'a' && input[i] <= 'z') || (input[i] >= 'A' &&
input[i] <= 'Z')) {
        i++;
        return 1;
    } else {
        return 0;
    }
}
```

OUTPUT



```
Recursive descent parsing for the following grammar

E -> TE'
E' -> +TE' | @
T -> FT'
T' -> *FT' | @
F -> (E) | ID

Enter the string to be checked: (a+b)*c

String is accepted

--------------------------------
Process exited after 23.01 seconds with return value 0
Press any key to continue . . .
```