Create a parse tree for the following language W --> id \* id + id

Given grammar is S --> T + S | T

T --> id \* T | id | (S)

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#include <stdbool.h>

#include <stdlib.h>

int isUpper(char c)

{

    if (c >= 'A'&& c <= 'Z')

        return1;

    else

        return0;

}

char \*subString(char \*str, int left, int right)

{

    int i;

    char \*subStr = (char \*)malloc(

        sizeof(char) \* (right - left + 2));

    for (i = left; i <= right; i++)

        subStr[i - left] = str[i];

    subStr[right - left + 1] = '\0';

    return (subStr);

}

int main()

{

    char production[10][10], s[200];

    char \*refStr;

    int cnt = 5, f = -1;

    strcpy(production[0], "S-T+S");

    strcpy(production[1], "T-c\*T");

    strcpy(production[2], "S-T");

    strcpy(production[3], "T-c");

    strcpy(production[4], "T-(S)");

    printf("\nOriginal Grammar: \n");

    for (int i = 0; i < cnt; i++)

    {

        printf("%s", production[i]);

        printf("\n");

    }

    printf("\nEnter Language : ");

    scanf("%[^\n]%\*c", s);

    refStr = subString(s, 2, strlen(s));

    printf("\n\nUpdates in main Production Grammar : \n");

    printf("%s", production[0]);

    for (int i = 1; i < cnt; i++)

    {

        char \*shortStr = subString(production[0], 2, strlen(production[0]));

        char modStr[30] = "";

        strcat(modStr, "S-");

        for (int k = 0; k < strlen(shortStr); k++)

        {

            if (shortStr[k] == production[i][0])

            {

                char \*useStr = subString(production[i], 2, strlen(production[i]));

                strcat(modStr, useStr);

            }

            else

            {

                strncat(modStr, &shortStr[k], 1);

            }

        }

        printf("\n%s", modStr);

        strcpy(production[0], modStr);

        char \*finStr = subString(modStr, 2, strlen(modStr));

        int terminate = 0;

        for (int r = 0; r < strlen(finStr); r++)

        {

            if (isUpper(finStr[r]))

            {

                terminate = 1;

                break;

            }

        }

        if (!strcmp(finStr, refStr) && terminate != 1)

        {

            f = 1;

            break;

        }

    }

    if (f == 1)

    {

        printf("\n\n%s language can be derived from the Original grammar.\n\n", s);

    }

    else

    {

        printf("\n\n%s language cannot be derived from the Original grammar.\n\n", s);

    }

    return0;

}