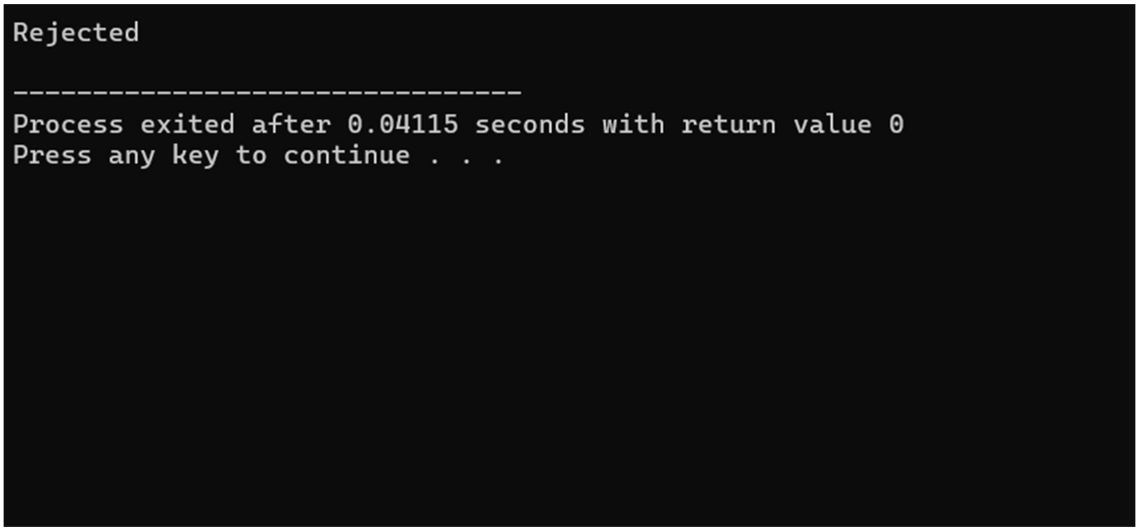
Problem Statement: Implement a simple DFA

Input:

```
//Simple DFA

#include<stdio.h>
#include<string.h>
#include<stdlib.h>
char *s="bbab";
int i=0;
void q0(char c);
void q1(char c);
void q2(char c);
void main()
{
        q0(s[i++]);
}
void q0(char c){
        switch(c){
                case 'a':
                        q1(s[i++]);
                case 'b':
                        q2(s[i++]);
                case '\0':
                        printf("Empty String\n");
                        exit(0);
                default:
                        printf("Invalid String\n");
                        exit(0);
        }
}
void q1(char c){
        switch(c){
                case 'a':
                        q1(s[i++]);
                case 'b':
                        q1(s[i++]);
                case '\0':
                        printf("Accepted\n");
                        exit(0);
                default:
                        printf("Invalid String\n");
                        exit(0);
        }
}

void q2(char c){
        switch(c){
                case 'a':
                        q2(s[i++]);
                case 'b':
                        q2(s[i++]);
                case '\0':
                        printf("Rejected\n");
                        exit(0);
                default:
                        printf("Invalid String\n");
                        exit(0);
        }
}
```

Output:



```
Rejected

--------------------------------
Process exited after 0.04115 seconds with return value 0
Press any key to continue . . .
```

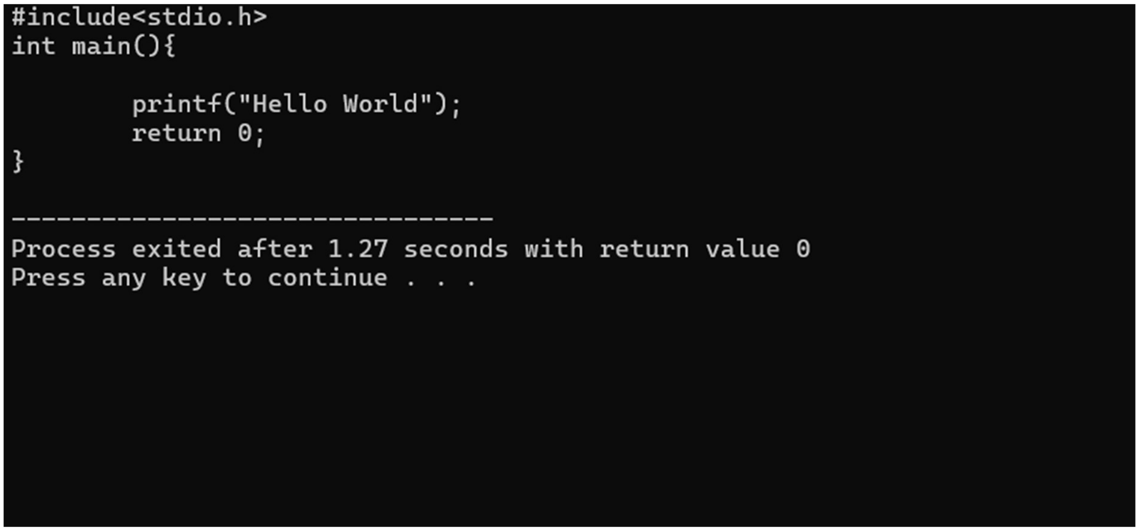
Problem Statement: Implement a C program to remove comments in a file.

Input:

```
//Remove Comments
#include<stdio.h>
#include<string.h>
void q0(char c);
void q1(char c);
void q2(char c);
void q3(char c);
void q4(char c);
```

```
FILE *fp;
char nextChar(){
    return fgetc(fp);
}
int main(){
    fp=fopen("h.c","r");
    q0(nextChar());
    return 0;
}
void q0(char c){
    switch(c){
        case '/':
            q1(nextChar());
            break;
        case EOF:
            break;
        default:
            printf("%c",c);
            q0(nextChar());
    }
}
void q1(char c){
    switch(c){
        case '/':
            q2(nextChar());
            break;
        case '*':
            q3(nextChar());
            break;
        case EOF:
            break;
        default:
            printf("%c",c);
            q0(nextChar());
    }
}
void q2(char c){
    switch(c){
        case '\n':
            q0(nextChar());
            break;
        case EOF:
            break;
        default:
            q2(nextChar());
    }
}
void q3(char c){
    switch(c){
        case '*':
            q4(nextChar());
            break;
        case EOF:
            break;
        default:
            q3(nextChar());
    }
}
void q4(char c){
    switch(c){
        case '/':
            q0(nextChar());
            break;
        case EOF:
            break;
        default:
            printf("%c",c);
            q3(nextChar());
    }
}
```

Output:



```
#include<stdio.h>
int main(){

        printf("Hello World");
        return 0;
}

--------------------------------
Process exited after 1.27 seconds with return value 0
Press any key to continue . . .
```

Problem Statement: Implement a C program to perform tokenization

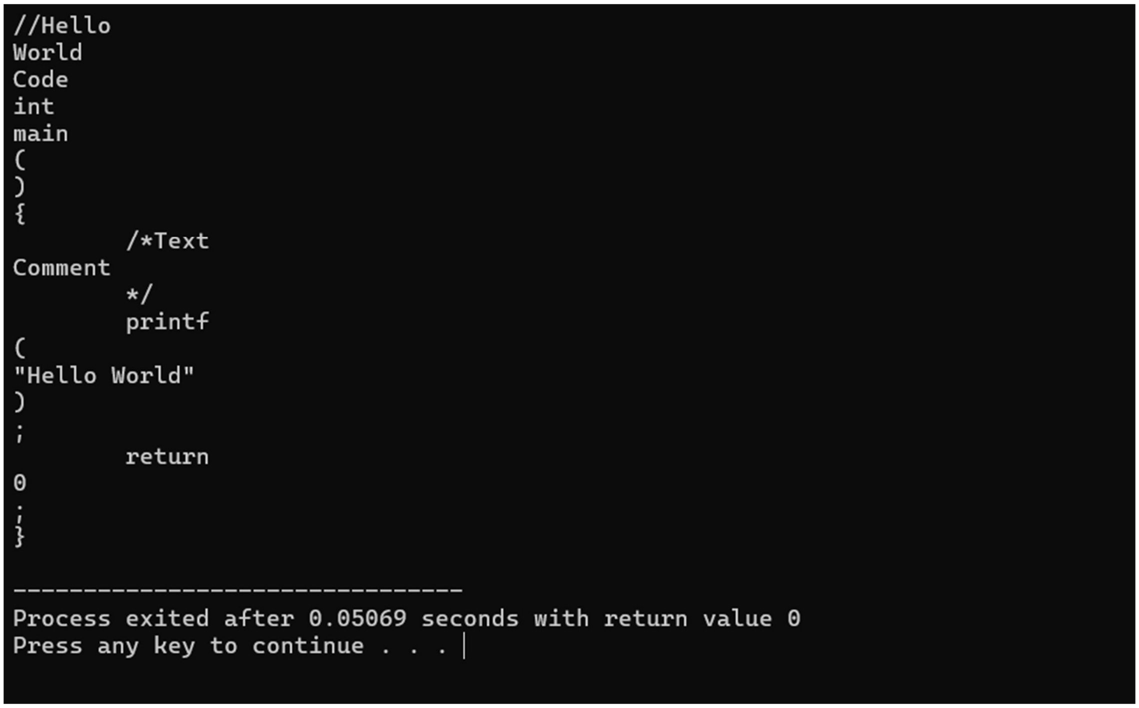
Input:

```
//Tokenization
#include<stdio.h>
#include<string.h>
void q0(char c);
void q1(char c);
void q2(char c);
void q3(char c);
FILE *fp;
char nextChar(){
	return fgetc(fp);
}
int main(){
	fp=fopen("h.c","r");
	q0(nextChar());
	return 0;
}
void q0(char c){
	switch(c){
		case ' ':
		case '\n':
			q0(nextChar());
			break;
		case '"':
			printf("%c",c);
			q3(nextChar());
			break;
		case '#':
			q1(nextChar());
			break;
		case EOF:
			break;
		default:
			printf("%c",c);
			q2(nextChar());
	}
}
void q1(char c){
	switch(c){
		case '\n':
			q0(nextChar());
			break;
		case EOF:
```

```
                        break;
                default:
                        q1(nextChar());
        }
}
void q2(char c){
        switch(c){
                case ' ':
                case '\n':
                        printf("\n");
                        q0(nextChar());
                        break;
                case '(':
                case '{':
                case ')':
                case '}':
                case ',':
                case ';':
                        printf("\n%c",c);
                        q2(nextChar());
                        break;
                case '"':
                        printf("\n%c",c);
                        q3(nextChar());
                        break;
                case EOF:
                        break;
                default:
                        printf("%c",c);
                        q2(nextChar());
        }
}
void q3(char c){
        switch(c){
                case '"':
                        printf("%c\n",c);
                        q0(nextChar());
                        break;
                case EOF:
                        break;
                default:
                        printf("%c",c);
                        q3(nextChar());
        }
```

}

Output:



```
//Hello
World
Code
int
main
(
)
{
        /*Text
Comment
        */
        printf
(
"Hello World"
)
;
        return
0
;
}

--------------------------------
Process exited after 0.05069 seconds with return value 0
Press any key to continue . . .
```

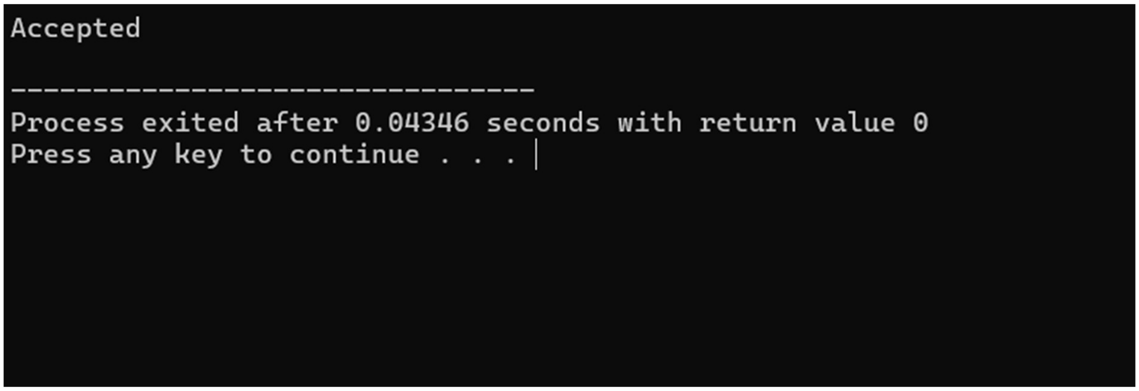
Problem Statement: Implement a C program to validate a password

Input:

```
//atleast and atmax one number in a password of length 4
#include<stdio.h>
#include<string.h>
void q0(int);
void q1(int);
void q2(int);
void q3(int);
void q4(int);
void q5(int);
void q6(int);
void q7(int);
char *str="lab44";
int i=0;
int isChar(int ch){
	if((ch>=65 && ch<=90) || (ch>=97 && ch<=122)){
		return 1;
	}
	return 0;
}
int isNo(int ch){
        if(ch>=48 && ch<=57){
                return 1;
        }
        return 0;
}
void main()
{
        int i=0;
        q0(i);
}
void q0(int i){
        char c=str[i];
        if(isChar(c)){
                q5(++i);
        }else if(isNo(c)){
                q1(++i);
        }else{
                printf("Invalid password");
        }
}
void q1(int i){
        char c=str[i];
        if(isChar(c)){
                q2(++i);
        }else{
                printf("Invalid password");
        }
}
void q2(int i){
        char c=str[i];
        if(isChar(c)){
                q3(++i);
        }else{
                printf("Invalid password");
        }
}
void q3(int i){
        char c=str[i];
        if(isChar(c)){
                q4(++i);
        }else{
                printf("Invalid password");
        }
}
```

```
void q4(int i){
        char c=str[i];
        if(c=='\0'){
                printf("Correct password\n");
        }else{
                printf("Invalid password\n");
        }
}
void q5(int i){
        char c=str[i];
        if(isChar(c)){
                q6(++i);
        }else if(isNo(c)){
                q2(++i);
        }else{
                printf("Invalid password\n");
        }
}
void q6(int i){
        char c=str[i];
        if(isChar(c)){
                q7(++i);
        }else if(isNo(c)){
                q3(++i);
        }else{
                printf("Invalid password\n");
        }
}
void q7(int i){
        char c=str[i];
        if(isNo(c)){
                q4(++i);
        }else{
                printf("Invalid password\n");
        }
}
```

Output:



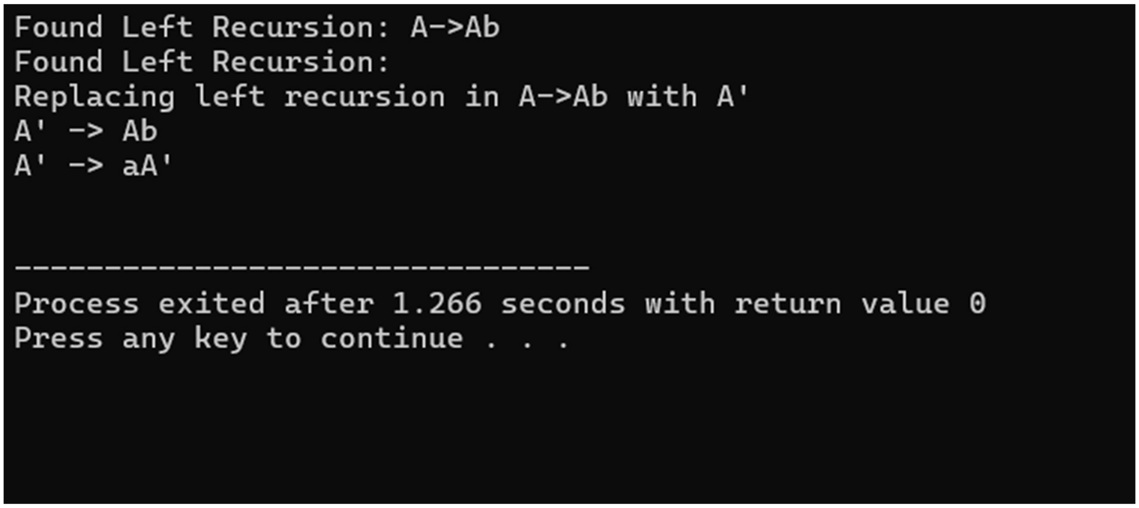
```
Accepted

--------------------------------
Process exited after 0.04346 seconds with return value 0
Press any key to continue . . .
```

Problem Statement: Implement a C program to find left recursion, left factoring and remove left factoring, left recursion

Input:

```
#include<stdio.h>
#include<string.h>
char *G[]={"A->Ab","A->a"};
int noOfProd=0;
void count(){
        while(G[++noOfProd]!='\0');
}
void findLeftRec();
void removeLeftRec();
void findLeftFact();
int main(){
        count();
        findLeftRec();
        removeLeftRec();
        findLeftFact();
        return 0;
}
void findLeftRec(){
        int i,j;
        for(i=0;i<noOfProd;i++){
                if(G[i][0]==G[i][3]){
                        printf("Found Left Recursion: %s\n",G[i]);
                }
                printf("Not Found Left Recursion: %s\n",G[i]);
        }
}
void removeLeftRec() {
   int i, j;
   char newNonTerminal[10];
   if (G[0][0] == G[0][3]) {
        strcpy(newNonTerminal, "A'");
        printf("Replacing left recursion in %s with %s\n", G[0], newNonTerminal);
        printf("%s -> %c%s\n", newNonTerminal, G[0][3], G[0] + 4);
        for (i = 1; i < noOfProd; i++) {
            if (G[i][0] == G[0][0]) {
                printf("%s -> %s%s\n", newNonTerminal, G[i] + 3, newNonTerminal);
            } else {
                printf("%s\n", G[i]);
            }
        }
    } else {
        printf("No left recursion found.\n");
    }
}
void findLeftFact(){
            int i,j;
            for(i=0;i<noOfProd;i++){
                        for(j=i+1;j<noOfProd;j++){
                                    if(G[i][3]==G[j][3]){
                                                printf("Found Left Factoring: %s\n",G[i]);
                                    }
                        }
            }
}
```

Output:



```
Found Left Recursion: A->Ab
Found Left Recursion:
Replacing left recursion in A->Ab with A'
A' -> Ab
A' -> aA'


--------------------------------
Process exited after 1.266 seconds with return value 0
Press any key to continue . . .
```

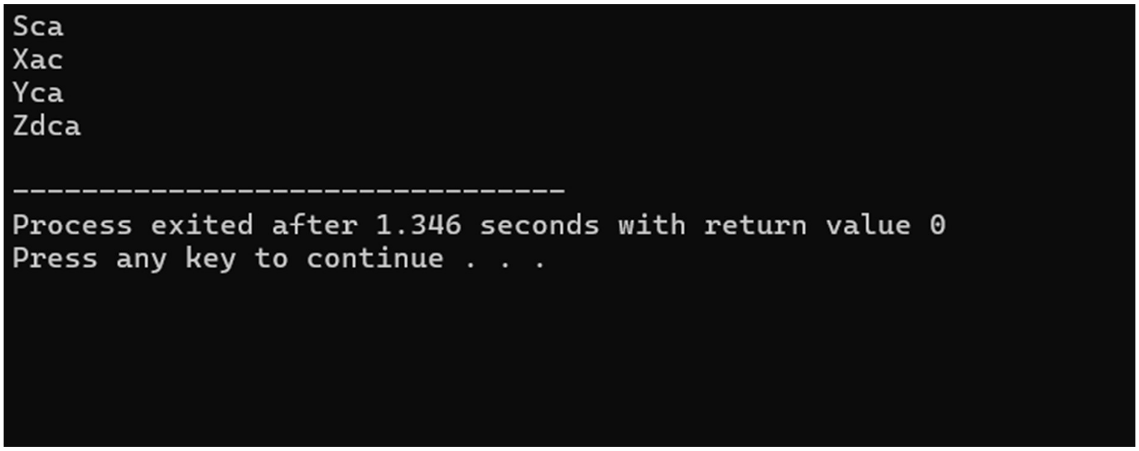
Problem Statement: Implement a C program to compute first and follow

Input:

```
//Compute the first and follow for the following:
#include<stdio.h>
#include<string.h>
int noOfProd=6;
int noOfVar=4;
char G[6][10]={"S->ZXY","S->XY","X->Xb","X->a","Y->cX","Z->d"};
char F[4][10]={"S","X","Y","Z"};
int isVar(char ch) {
  if (ch >= 'A' && ch <= 'Z')
    return 1;
  return 0;
}
void add(int var,char ter){
        int i=1;
        while(F[var][i]!=ter && F[var][i]!='\0')
                i++;
        if(F[var][i]=='\0'){
                F[var][i]=ter;
                F[var][i+1]='\0';
        }
}
void Union(int i,char var){
        int j=0,k=1;
        while(F[j][0]!=var)
                j++;
        while(F[j][k]!='\0'){
                add(i,F[j][k]);
                k++;
        }
}
int main() {
  int i, j;
  for (i = 0; i < noOfVar; i++) {
    for (j = 0; j < noOfProd; j++) {
      if (F[i][0] == G[j][0] && !isVar(G[j][3])) {
        add(i, G[j][3]);
      }
    }
  }
  for (i = 0; i < noOfVar; i++) {
    for (j = 0; j < noOfProd; j++) {
      int len = strlen(G[j]);
```

```
            if (isVar(G[j][len - 1])) {
                Union(i, G[j][len - 1]);
            }
        }
    }
    for (i = 0; i < noOfVar; i++) {
        printf("%s\n", F[i]);
    }
    return 0;
}
```

Output:



```
Sca
Xac
Yca
Zdca

--------------------------------
Process exited after 1.346 seconds with return value 0
Press any key to continue . . .
```