

**Name :** Rajvardhan Reddy  
**Reg No :** 180905093  
**Sec :** B  
**Roll No:** 19

### **CD LAB -1 : Basic File Handling Operations**

**P1)** Write a C program to copy the contents of source file to destination file

**Program:**

```
// Program to copy contents of source file to destination file
#include <stdio.h>
#include <stdlib.h> // For exit()
int main()
{
    FILE *fptr1, *fptr2;
    char filename[100], c;
    printf("Enter the filename to open for reading: \n");
    scanf("%s", filename);
    fptr1 = fopen(filename, "r");
    // Open one file for reading
    if (fptr1 == NULL)
    {
        printf("Cannot open file %s \n", filename);
        exit(0);
    }
    printf("Enter the filename to open for writing: \n");
    scanf("%s", filename);
    fptr2 = fopen(filename, "w+"); // Open another file for writing
    c = fgetc(fptr1);
    // Read contents from file
    while (c != EOF)
    {
        fputc(c, fptr2);
        c = fgetc(fptr1);
    }
    printf("\nContents copied to %s", filename);
    fclose(fptr1);
    fclose(fptr2);
    return 0;
}
```

### Sample Input and Output :

```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab
1$ cat destination.txt
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab
1$ cc lab1_p1.c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab
1$ ./a.out
Enter the filename to open for reading:
source.txt
Enter the filename to open for writing:
destination.txt

Contents copied to destination.txt
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab
1$ cat destination.txt
This file is for demonstartion of compiler design lab 1 program
```

**P2)** To count the number of lines and characters in a file.

#### **Program :**

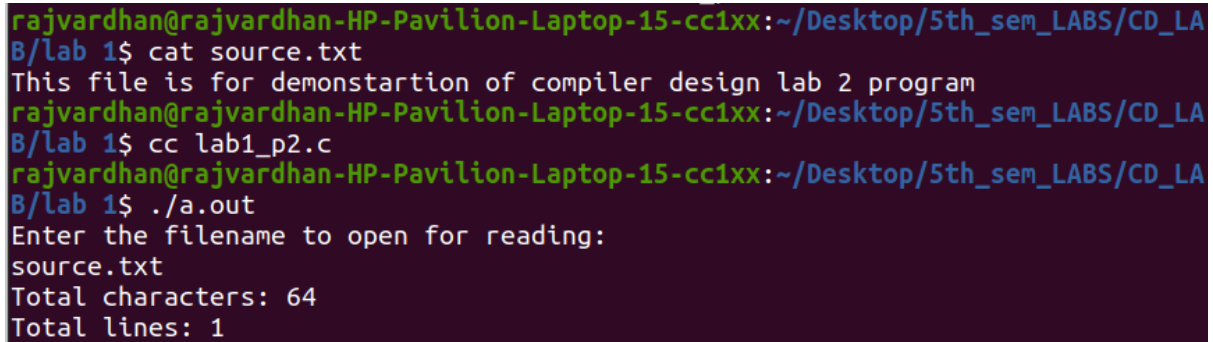
```
#include <stdio.h>
#include <stdlib.h> // For exit()
int main()
{
    FILE *fp;
    char filename[100], c;
    int lines = 0;
    int characters = 0;
    printf("Enter the filename to open for reading: \n");
    scanf("%s", filename);
    fp = fopen(filename, "r");
    if (fp == NULL)
    {
        printf("Cannot open file %s \n", filename);
        exit(0);
    }
    c = fgetc(fp);
    while (c != EOF)
    {
        characters++;
        if (c == '\n')
        {
            lines++;
        }
        c = fgetc(fp);
    }
```

```

printf("Total characters: %d\n", characters);
printf("Total lines: %d\n", lines);
fclose(fp);
return 0;
}

```

### Output :



```

rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab 1$ cat source.txt
This file is for demonstration of compiler design lab 2 program
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab 1$ cc lab1_p2.c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab 1$ ./a.out
Enter the filename to open for reading:
source.txt
Total characters: 64
Total lines: 1

```

**P3)** To reverse the file contents and store in another file. Also display the size of file using file handling function.

### Program :

```

#include <stdio.h>
#include <stdlib.h> // For exit()
int main()
{
FILE *fp,*fp2;
char filename[100], c;
printf("Enter the filename to open for reading: \n");
scanf("%s", filename);
fp = fopen(filename, "r");
if (fp == NULL)
{
printf("Cannot open file %s \n", filename);
exit(0);
}
printf("Enter the filename to open for writing: \n");
scanf("%s", filename);
fp2 = fopen(filename, "w+");
fseek(fp, 0, SEEK_END);
int f1Size = ftell(fp);
printf("Size of file: %d bytes\n", f1Size);
fseek(fp, f1Size-1, SEEK_SET);
fseek(fp2, 0, SEEK_SET);

```

```

for (int i = 0; i < f1Size-1; i++)
{
c = fgetc(fp);
if (c == EOF)
{
printf("Breaking: %d\n", i);
break;
}
fputc(c, fp2);
fseek(fp, f1Size-(i+2), SEEK_SET);
}
printf("File content reversed and stored in destination.\n");
fclose(fp);
fclose(fp2);
return 0;
}

```

### Output :

```

rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab 1$ cat source.txt
This file is for demonstartion of compiler design lab 3 program
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab 1$ cc lab1_p3.c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab 1$ ./a.out
Enter the filename to open for reading:
source.txt
Enter the filename to open for writing:
destination.txt
Size of file: 64 bytes
File content reversed and stored in destination.
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab 1$ cat destination.txt
margorp 3 bal ngised relipmoc fo noitratsnomed rof si elif sihrajvardhan@rajva
rdhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab 1$

```

**P4)** That merges lines alternatively from 2 files and stores it in a resultant file.

### Program :

```

#include <stdio.h>
#include <stdlib.h> // For exit()
int main()
{
FILE *f1,*f2,*f3;
char fn1[100], fn2[100], fn3[100], c;
printf("Enter the file1, file2, file3 : \n");

```

```
scanf("%s", fn1);
scanf("%s", fn2);
scanf("%s", fn3);
f1 = fopen(fn1, "r");
f2 = fopen(fn2, "r");
f3 = fopen(fn3, "w+");
if (f1 == NULL || f2 == NULL)
{
printf("Cannot open files.\n");
exit(0);
}
char ch1, ch2;
ch1 = ch2 = 'a';
while (1)
{
if (ch1 != EOF)
{
ch1 = fgetc(f1);
while (ch1 != '\n')
{
if (ch1 == EOF)
break;
fputc(ch1, f3);
ch1 = fgetc(f1);
}
fputc('\n', f3);
}
if (ch2 != EOF)
{
ch2 = fgetc(f2);
while (ch2 != '\n')
{
if (ch2 == EOF)
break;
fputc(ch2, f3);
ch2 = fgetc(f2);
}
fputc('\n', f3);
}
if (ch1 == EOF && ch2 == EOF)
break;
}
printf("Done");
fclose(f1);
fclose(f2);
```

```
fclose(f3);
return 0;
}
```

### Output :

```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/Lab 1$ cc lab1_p4.c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/Lab 1$ ./a.out
Enter the file1, file2, file3 :
source.txt
source-2.txt
destination.txt
Done
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/Lab 1$ cat destination.txt
This file is for demonstartion of compiler design lab 4 program - file 1
This file is for demonstartion of compiler design lab 4 program - file 2
```

## CD LAB -2 : PRELIMINARY SCANNING APPLICATIONS

**P1)** Program to remove single and multiline comments from a given 'C' file.

### Program :

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    FILE *fa,
    *fb;
    int ca, cb;
    fa = fopen("demo_lab2_p1.c", "r");
    if (fa == NULL)
    {
        printf("Cannot open file \n");
        exit(0);
    }
    fb = fopen("demo_lab2_p1_out.c", "w");
    ca = getc(fa);
    while (ca != EOF)
    {
        if (ca == ' ')
        {
            putc(ca, fb);
            while (ca == ' ')
            ca = getc(fa);
        }
    }
}
```

```
}
if (ca == '/')
{
cb = getc(fa);
if (cb == '/')
{
while (ca != '\n')
ca = getc(fa);
}
else if (cb == '*')
{
do
{
while (ca != '*')
ca = getc(fa);
ca = getc(fa);
} while (ca != '/');
}
else
{
putc(ca, fb);
putc(cb, fb);
}
}
else
{
putc(ca, fb);
}
ca = getc(fa);
}
fclose(fa);
fclose(fb);
return 0;
}
```

## Sample Input and Output :

```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab 2$ cat demo_lab2_p1.c
// This is a single line comment
/* *****This is a
*****Multiline Comment
**** */
#include <stdio.h>
void main()
{
FILE *fopen(), *fp;
int c ;
fp = fopen( "prog.c", "r" ); //Comment
12c = getc( fp ) ;
while ( c
!=
EOF )
{
putchar( c );
c = getc ( fp );
}
/*multiline
comment */
fclose(
fp );
}rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab 2$ cc lab2_p1.c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab 2$ ./a.out demo_lab2_p1.c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab 2$ cat demo_lab2_p1_out.c

#include <stdio.h>
void main()
{
FILE *fopen(), *fp;
int c ;
fp = fopen( "prog.c", "r" ); 12c = getc( fp ) ;
while ( c
!=
EOF )
{
putchar( c );
c = getc ( fp );
}

fclose(
fp );
```

**P2)** That takes a file as input and replaces blank spaces and tabs by single space and writes the output to a file.

### Program :

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

```
int main()
{
    FILE *fa, *fb;
    char input[256];
    printf("Enter file name: ");
    scanf("%s",input);
    fa = fopen(input, "r");
    fb = fopen("output.txt", "w");
    if(fa==NULL || fb==NULL) printf("Invalid files\n");
    else
    {
```

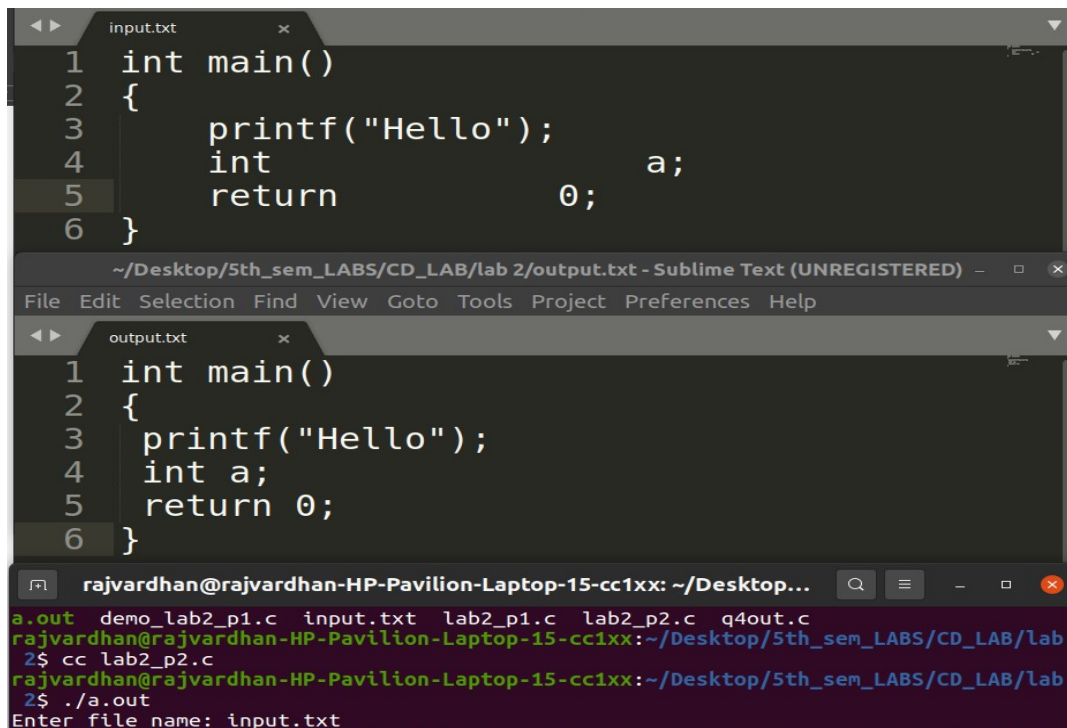


```

    int c;
    while((c = getc(fa))!=EOF)
    {
        if(c==' ' || c=='\t')
        {
            while(c==' ' || c=='\t') c= getc(fa);
            putc(' ',fb);
        }
        putc(c,fb);
    }
    fclose(fa);
    fclose(fb);
}
return 0;
}

```

**Output :**



The screenshot shows a code editor with two files: `input.txt` and `output.txt`. Both files contain the same C code for a `main` function that prints "Hello". The terminal window shows the compilation and execution of the program.

```

1 int main()
2 {
3     printf("Hello");
4     int a;
5     return 0;
6 }

```

```

~/Desktop/5th_sem_LABS/CD_LAB/lab 2/output.txt - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

```

```

1 int main()
2 {
3     printf("Hello");
4     int a;
5     return 0;
6 }

```

```

rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx: ~/Desktop...
a.out demo_lab2_p1.c input.txt lab2_p1.c lab2_p2.c q4out.c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab
2$ cc lab2_p2.c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab
2$ ./a.out
Enter file name: input.txt

```

**P3)** To discard preprocessor directives from the given input 'C' file.

**Program :**

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

const char *directives[] = {"#include", "#define", "#if"};

int is_directive(char *str)
{
    for(int i = 0; i < sizeof(directives)/sizeof(char *); i++)
    {
        int len = strlen(directives[i]);
        if(strncmp(str, directives[i], len) == 0)
            return 1;
    }
    return 0;
}

int main()
{
    FILE *fa,*fb;
    char input[256];
    printf("Enter file name: ");
    scanf("%s",input);

    fa = fopen(input, "r");
    fb = fopen("output.c", "w");

    if(fa == NULL || fb == NULL)
    {
        perror("Invalid files\n");
        return 1;
    }
    char line[256];
    while(fgets(line, 256, fa))
    {
        if(!is_directive(line))
        {
            fputs(line, fb);
        }
    }
    fclose(fa);
```

```

fclose(fb);
}

```

**Output :**

```

input.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #define multiply(f1,f2) (f1*f2)
4
5  int main()
6  {
7      printf("Hello");
8      return 0;
9  }

~/Desktop/5th_sem_LABS/CD_LAB/lab 2/output.c - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

output.c
1
2  int main()
3  {
4      printf("Hello");
5      return 0;
6  }

rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx: ~/De...
demo_lab2_p1.c  input.c          lab2_p1.c  lab2_p3.c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LA
B/Lab 2$ cc lab2_p3.c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LA
B/Lab 2$ ./a.out
Enter file name: input.c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LA
B/Lab 2$ ls
a.out      demo_lab2_p1_out.c  input.txt  lab2_p2.c  output.c
demo_lab2_p1.c  input.c          lab2_p1.c  lab2_p3.c  output.txt

```

**P4)** That takes C program as input, recognizes all the keywords and prints them in upper case.

**Program :**

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stddef.h>

```

```

const char *keywords[33] = {"auto", "double", "int", "struct", "break", "else", "long",
"switch", "case", "enum", "register", "typedef", "char", "extern", "return", "union",
"continue", "for", "signed", "void", "do", "if", "static", "while", "default", "goto",
"sizeof", "volatile", "const", "float", "short", "unsigned", "printf"};

```

```

int isKeyword (char *word)

```

```

{
    for (int i=0; i<33; i++)
    {
        if (strcmp(word, keywords[i]) == 0)
            return 1;
    }
    return 0;
}

int main (int argc, const char * argv [])
{
    printf("Enter file name: ");
    char input[256];
    scanf("%s",input);
    FILE *f = fopen(input, "r");
    if (f == NULL)
    {
        printf("Cannot open file\n");
        exit(0);
    }

    char buffer[1024];
    const char delimiters[] = " .,:;!?-_(){}[]\n\t";

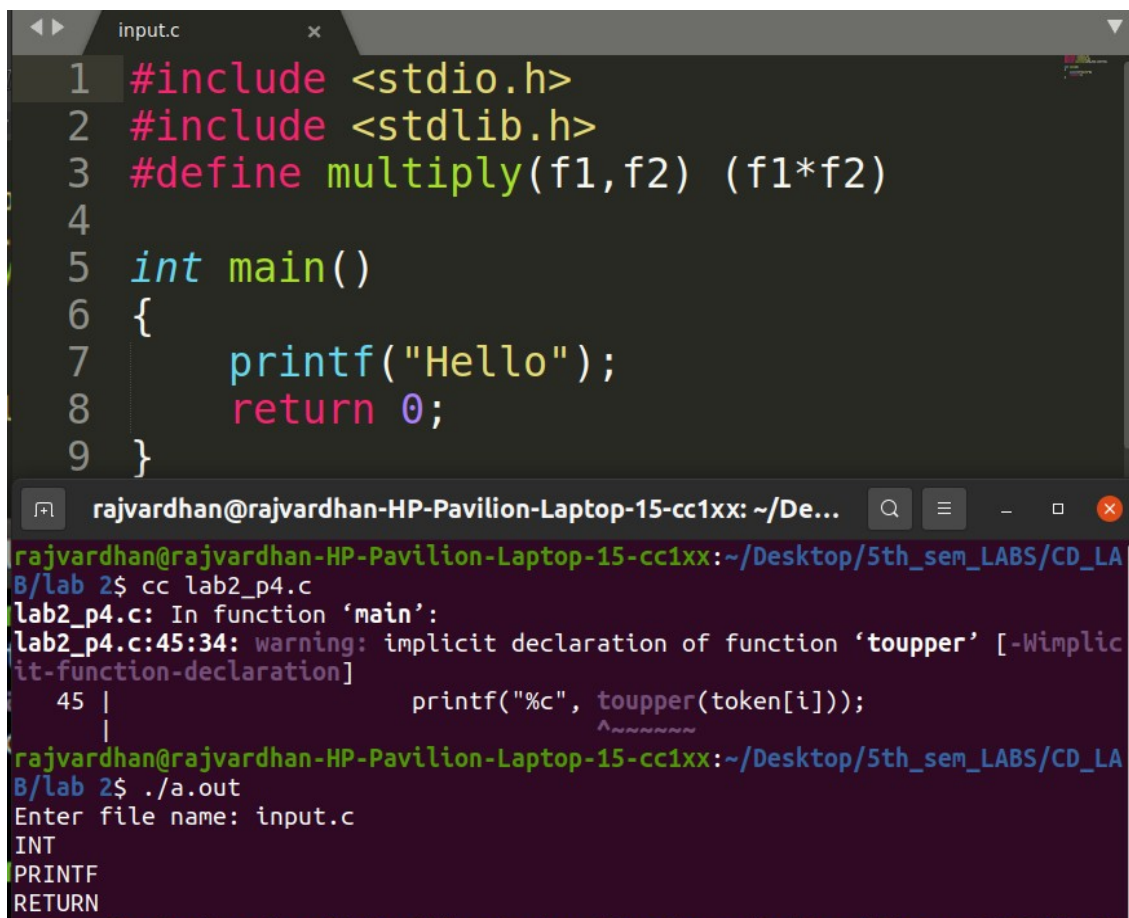
    while (fgets(buffer, 1024, f))
    {
        char *cp = (char *)malloc(1024 * sizeof(char));
        strcpy(cp, buffer);

        char *token = (char *)malloc(256 * sizeof(char));

        while((token = strsep(&cp, delimiters)))
        {
            if(isKeyword(token))
            {
                for (int i=0; i<strlen(token); i++)
                    printf("%c", toupper(token[i]));
                printf("\n");
            }
        }
    }
    fclose(f);
    return 0;
}

```

**Output :**



The screenshot shows a code editor window titled 'input.c' with the following C code:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define multiply(f1,f2) (f1*f2)
4
5 int main()
6 {
7     printf("Hello");
8     return 0;
9 }
```

Below the code editor is a terminal window. The prompt is 'rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx: ~/De...'. The user enters 'cc lab2\_p4.c'. The terminal output shows a warning: 'lab2\_p4.c: In function 'main': lab2\_p4.c:45:34: warning: implicit declaration of function 'toupper' [-Wimplicit-function-declaration]'. The user then enters './a.out'. The terminal output shows 'Enter file name: input.c', 'INT', 'PRINTF', and 'RETURN'.

## CD LAB - 3 : Construction of Token Generator

**P1)** Write a program in 'C' to identify the arithmetic and relational operators from the given input 'C' file.

**Program :**

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int main()
{
    char c,buf[10];
    FILE *fp=fopen("digit.c","r");
    c = fgetc(fp);
    if (fp == NULL)
    {
```

```

printf("Cannot open file \n");
exit(0);
}
while(c!=EOF)
{
int i=0;
buf[0]='\0';
if(c=='=')
{
buf[i++]=c;
c = fgetc(fp);
if(c=='=')
{
buf[i++]=c;
buf[i]='\0';
printf("\n Relational operator : %s",buf);
}
else
{
buf[i]='\0';
printf("\n Assignment operator: %s ",buf);
}
}
else
{
if(c=='<'||c=='>'||c=='!')
{
buf[i++]=c;
c = fgetc(fp);
if(c=='=')
{
buf[i++]=c;
}
buf[i]='\0';
printf("\n Relational operator : %s",buf);
}
else
{
buf[i]='\0';
}
}
c = fgetc(fp);
} }

```

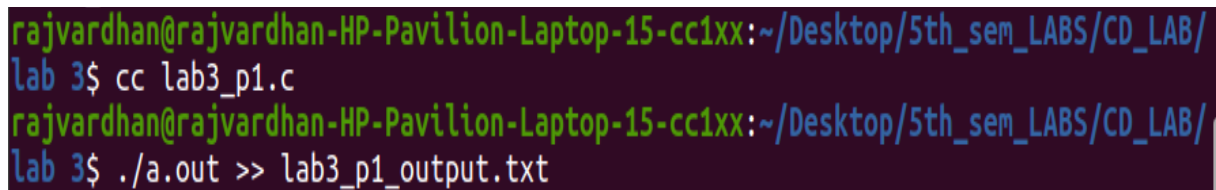
**input file :**

```

#include <stdio.h>
int main(int argc, char const *argv[])
{
    int a[100],b=0,c=0,i=0;
    for(i=0;i<100;i++){
        a[i] = i+1;
    }
    for (i=0;i<100;i++){
        if(a[i]%2==0){
            b+=a[i];
        }
        else c+=a[i];
    }
    printf("The even and odd sums of first 100 natural numbers are: %d, %d\n",b,c);
    return 0;
}

```

### Output :



```

rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/
lab 3$ cc lab3_p1.c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/
lab 3$ ./a.out >> lab3_p1_output.txt

```

### Output Text File :

```

Relational operator : <
Relational operator : >
Assignment operator: =
Assignment operator: =
Assignment operator: =
Assignment operator: =
Relational operator : <
Assignment operator: =
Assignment operator: =
Relational operator : <
Relational operator : ==
Assignment operator: =
Assignment operator: =

```

**P2)** Design a lexical analyzer which contains getNextToken( ) for a simple C program to create a structure of token each time and return, which includes row number, column number and token type. The tokens to be identified are arithmetic operators, relational operators, logical operators, special symbols, keywords, numerical constants, string literals and identifiers. Also, getNextToken() should ignore all the tokens when encountered inside single line or multiline comment or inside string literal. Preprocessor directive should also be stripped.

**Program :**

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

struct token
{
    char lexeme[64];
    unsigned int row, col;
    char type[20];
};

static int row = 1, col = 1;
char buf[2048];

char specialsymbols[] = {'?', ';', ':', ',', '.'};

const char *keywords[] = {"const", "bool", "char", "int", "float", "double",
"unsigned", "return", "for", "while", "do", "switch", "if", "else", "case", "break",
"printf", "continue"};

char arithmeticsymbols[] = {'*', '%'};

int isKeyword(char *word)
{
    for (int i = 0; i < sizeof(keywords) / sizeof(char *); i++)
    {
        if (strcmp(word, keywords[i]) == 0)
            return 1;
    }
    return 0;
}
```



```

int charBelongsTo(int c, char *arr)
{
    int len = 0;
    if (arr == specialsymbols)
        len = sizeof(specialsymbols) / sizeof(char);
    else if (arr == arithmeticsymbols)
        len = sizeof(arithmeticsymbols) / sizeof(char);

    for (int i = 0; i < len; i++)
    {
        if (c == arr[i])
            return 1;
    }
    return 0;
}

void fillToken(struct token *t, char c, int row, int col, char *type)
{
    t->lexeme[0] = c;
    t->lexeme[1] = '\0';
    t->row = row;
    t->col = col;
    strcpy(t->type, type);
}

void newLine()
{
    row++;
    col = 1;
}

struct token getNextToken(FILE *f)
{
    int c;
    struct token tkn =
    {
        .row = -1};

    int gotToken = 0;
    while (!gotToken && (c = fgetc(f)) != EOF)
    {
        if (charBelongsTo(c, specialsymbols))
        {
            fillToken(&tkn, c, row, col, "SS");
            gotToken = 1;

```

```

        ++col;
    }
    else if (charBelongsTo(c, arithmeticsymbols))
    {
        fillToken(&tkn, c, row, col, "ARITHMETIC OPERATOR");
        gotToken = 1;
        ++col;
    }
    else if (c == '(')
    {
        fillToken(&tkn, c, row, col, "LB");
        gotToken = 1;
        ++col;
    }
    else if (c == ')')
    {
        fillToken(&tkn, c, row, col, "RB");
        gotToken = 1;
        ++col;
    }
    else if (c == '{')
    {
        fillToken(&tkn, c, row, col, "LC");
        gotToken = 1;
        ++col;
    }
    else if (c == '}')
    {
        fillToken(&tkn, c, row, col, "RC");
        gotToken = 1;
        ++col;
    }
    else if (c == '[')
    {
        fillToken(&tkn, c, row, col, "LS");
        gotToken = 1;
        ++col;
    }
    else if (c == ']')
    {
        fillToken(&tkn, c, row, col, "RS");
        gotToken = 1;
        ++col;
    }
    else if (c == '+')

```

```

{
    int d = fgetc(f);
    if (d != '+')
    {
        fillToken(&tkn, c, row, col, "ARITHMETIC OPERATOR");
        gotToken = 1;
        ++col;
        fseek(f, -1, SEEK_CUR); //go back 1 step *
    }
    else
    {
        fillToken(&tkn, c, row, col, "UNARY OPERATOR");
        strcpy(tkn.lexeme, "++");
        gotToken = 1;
        col += 2; //skip next as it is already included
    }
}
else if (c == '-')
{
    int d = fgetc(f);
    if (d != '-')
    {
        fillToken(&tkn, c, row, col, "ARITHMETIC OPERATOR");
        gotToken = 1;
        ++col;
        fseek(f, -1, SEEK_CUR); //go back 1 step *
    }
    else
    {
        fillToken(&tkn, c, row, col, "UNARY OPERATOR");
        strcpy(tkn.lexeme, "--");
        gotToken = 1;
        col += 2; //skip next as it is already included
    }
}
else if (c == '=')
{
    int d = fgetc(f);
    if (d != '=')
    {
        fillToken(&tkn, c, row, col, "ASSIGNMENT OPERATOR");
        gotToken = 1;
        ++col;
        fseek(f, -1, SEEK_CUR); //go back 1 step *
    }
}

```

```

else
{
    fillToken(&tkn, c, row, col, "RELATIONAL OPERATOR");
    strcpy(tkn.lexeme, "==");
    gotToken = 1;
    col += 2; //skip next as it is already included
}
}
else if (isdigit(c))
{
    tkn.row = row;
    tkn.col = col++;
    tkn.lexeme[0] = c;
    int k = 1;
    while ((c = fgetc(f)) != EOF && isdigit(c))
    {
        tkn.lexeme[k++] = c;
        col++;
    }
    tkn.lexeme[k] = '\0';
    strcpy(tkn.type, "NUMBER");
    gotToken = 1;
    fseek(f, -1, SEEK_CUR); //go back 1 step *
}
else if (c == '#')
{
    while ((c = fgetc(f)) != EOF && c != '\n')
        ;
    newLine();
}
else if (c == '\n')
{
    newLine();
    c = fgetc(f);
    if (c == '#')
    {
        while ((c = fgetc(f)) != EOF && c != '\n')
            ;
        newLine();
    }
    else if (c != EOF)
        fseek(f, -1, SEEK_CUR);
}
else if (isspace(c))
    ++col;

```

```

else if (isalpha(c) || c == '_')
{
    tkn.row = row;
    tkn.col = col++;
    tkn.lexeme[0] = c;
    int k = 1;
    while ((c = fgetc(f)) != EOF && isalnum(c))
    {
        tkn.lexeme[k++] = c;
        ++col;
    }
    tkn.lexeme[k] = '\0';
    if (isKeyword(tkn.lexeme))
        strcpy(tkn.type, "KEYWORD");
    else
        strcpy(tkn.type, "IDENTIFIER");
    gotToken = 1;
    fseek(f, -1, SEEK_CUR);
}
else if (c == '/')
{
    int d = fgetc(f);
    ++col;
    if (d == '/')
    {
        while ((c = fgetc(f)) != EOF && c != '\n')
            ++col;
        if (c == '\n')
            newLine();
    }
    else if (d == '*')
    {
        do
        {
            if (d == '\n')
                newLine();
            while ((c = fgetc(f)) != EOF && c != '*')
            {
                ++col;
                if (c == '\n')
                    newLine();
            }
            ++col;
        } while ((d = fgetc(f)) != EOF && d != '/' && (++col));
        ++col;
    }
}

```

```

    }
    else
    {
        fillToken(&tkn, c, row, --col, "ARITHMETIC OPERATOR");
        gotToken = 1;
        fseek(f, -1, SEEK_CUR);
    }
}
else if (c == "")
{
    tkn.row = row;
    tkn.col = col;
    strcpy(tkn.type, "STRING LITERAL");
    int k = 1;
    tkn.lexeme[0] = "";
    while ((c = fgetc(f)) != EOF && c != "")
    {
        tkn.lexeme[k++] = c;
        ++col;
    }
    tkn.lexeme[k] = "";
    gotToken = 1;
}
else if (c == '<' || c == '>' || c == '!')
{
    fillToken(&tkn, c, row, col, "RELATIONAL OPERATOR");
    ++col;
    int d = fgetc(f);
    if (d == '=')
    {
        ++col;
        strcat(tkn.lexeme, "=");
    }
    else
    {
        if (c == '!')
            strcpy(tkn.type, "LOGICAL OPERATOR");
        fseek(f, -1, SEEK_CUR);
    }
    gotToken = 1;
}
else if (c == '&' || c == '|')
{
    int d = fgetc(f);
    if (c == d)

```

```

    {
        tkn.lexeme[0] = tkn.lexeme[1] = c;
        tkn.lexeme[2] = '\0';
        tkn.row = row;
        tkn.col = col;
        ++col;
        gotToken = 1;
        strcpy(tkn.type, "LOGICAL OPERATOR");
    }
    else
    {
        tkn.lexeme[0] = c;
        tkn.lexeme[1] = '\0';
        tkn.row = row;
        tkn.col = col;
        ++col;
        gotToken = 1;
        strcpy(tkn.type, "BITWISE OPERATOR");
        fseek(f, -1, SEEK_CUR);
    }
    ++col;
}
else
    ++col;
}
return tkn;
}

int main()
{
    printf("Enter file name: ");
    char input[256];
    scanf("%s", input);
    FILE *f = fopen(input, "r");
    if (f == NULL)
    {
        printf("Cannot open file\n");
        exit(0);
    }

    struct token t;
    while ((t = getNextToken(f)).row != -1)
        printf("<%s, %d, %d, %s>\n", t.lexeme, t.row, t.col, t.type);

    fclose(f);

```

```
    return 0;
}
```

**input file :**

```
#include<stdio.h>
#include<stdlib.h>
int main(){
    int p,q,m,n;
    printf("Enter the dimensions of the first matrix: ");
    scanf("%d%d",&p,&q);
    printf("Enter the dimensions of the second matrix: ");
    scanf("%d%d",&m,&n);
    if(m!=q){
        printf("The matrices are unsuitable for multiplication");
        return 0;
    }
    int size1 = p*q*sizeof(int);
    int size2 = m*n*sizeof(int);
    int arr1 = (int)malloc(size1);
    int arr2 = (int)malloc(size2);
    printf("Enter the elements of the first array\n");
    int i,j,k;
    for (i=0;i<p;i++){
        for(j=0;j<q;j++){
            scanf("%d",arr1+q*i+j);
        }
    }
    printf("Enter the elements of the second array\n");
    for (i=0;i<m;i++){
        for(j=0;j<n;j++){
            scanf("%d",arr2+n*i+j);
        }
    }
    int size3 = p*n*sizeof(int);
    int res = (int)malloc(size3);
    int sum = 0;
    for(i=0;i<p;i++){
        for(j=0;j<n;j++){
            for(k=0;k<m;k++){
                sum+= arr1[i*m+k] * arr2[k*n+j];
            }
            res[i*n+j] = sum;
            sum=0;
        }
    }
}
```

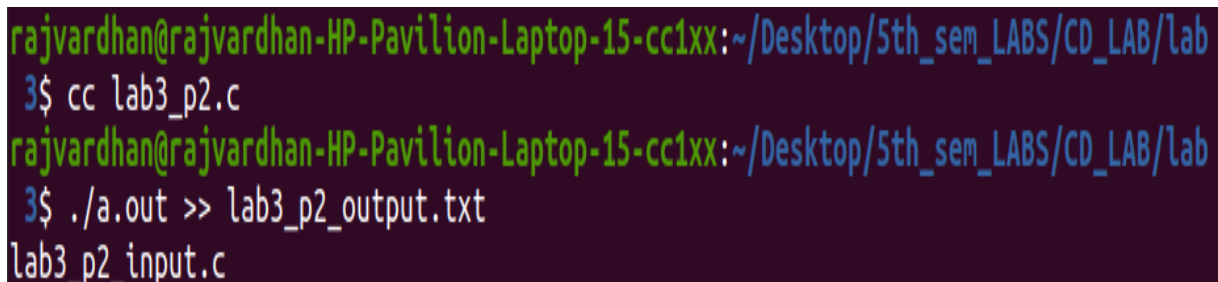


```

        printf("The result is\n");
        for(i=0;i<p;i++){
            for(j=0;j<n;j++){
                printf("%d ",*(res+n*i+j));
            }
            printf("\n");
        }
        return 0;
}

```

### Output :



```

rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab
3$ cc lab3_p2.c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab
3$ ./a.out >> lab3_p2_output.txt
lab3 p2 input.c

```

### Output Text-File :

```

<int, 3, 1, KEYWORD>
<main, 3, 5, IDENTIFIER>
<(, 3, 9, LB>
<), 3, 10, RB>
<{, 3, 11, LC>
<int, 4, 2, KEYWORD>
<p, 4, 6, IDENTIFIER>
<,, 4, 7, SS>
<q, 4, 8, IDENTIFIER>
<,, 4, 9, SS>
<m, 4, 10, IDENTIFIER>
<,, 4, 11, SS>
<n, 4, 12, IDENTIFIER>
<;, 4, 13, SS>
<printf, 5, 2, KEYWORD>
<(, 5, 8, LB>
<"Enter the dimensions of the first matrix: ", 5, 9, STRING LITERAL>
<), 5, 51, RB>
<;, 5, 52, SS>
<scanf, 6, 2, IDENTIFIER>
<(, 6, 7, LB>
<"%d%d", 6, 8, STRING LITERAL>

```

<,, 6, 12, SS>  
<&, 6, 13, BITWISE OPERATOR>  
<p, 6, 15, IDENTIFIER>  
<,, 6, 16, SS>  
<&, 6, 17, BITWISE OPERATOR>  
<q, 6, 19, IDENTIFIER>  
<), 6, 20, RB>  
<;, 6, 21, SS>  
<printf, 7, 2, KEYWORD>  
<(, 7, 8, LB>  
<"Enter the dimensions of the second matrix: ", 7, 9, STRING LITERAL>  
<), 7, 52, RB>  
<;, 7, 53, SS>  
<scanf, 8, 2, IDENTIFIER>  
<(, 8, 7, LB>  
<"%d%d", 8, 8, STRING LITERAL>  
<,, 8, 12, SS>  
<&, 8, 13, BITWISE OPERATOR>  
<m, 8, 15, IDENTIFIER>  
<,, 8, 16, SS>  
<&, 8, 17, BITWISE OPERATOR>  
<n, 8, 19, IDENTIFIER>  
<), 8, 20, RB>  
<;, 8, 21, SS>  
<if, 9, 2, KEYWORD>  
<(, 9, 4, LB>  
<m, 9, 5, IDENTIFIER>  
<!=, 9, 6, RELATIONAL OPERATOR>  
<q, 9, 8, IDENTIFIER>  
<), 9, 9, RB>  
<{, 9, 10, LC>  
<printf, 10, 3, KEYWORD>  
<(, 10, 9, LB>  
<"The matrices are unsuitable for multiplication", 10, 10, STRING LITERAL>  
<), 10, 56, RB>  
<;, 10, 57, SS>  
<return, 11, 3, KEYWORD>  
<0, 11, 10, NUMBER>  
<;, 11, 11, SS>  
<}, 12, 2, RC>  
<int, 13, 2, KEYWORD>  
<size1, 13, 6, IDENTIFIER>  
<=, 13, 12, ASSIGNMENT OPERATOR>  
<p, 13, 14, IDENTIFIER>  
<\*, 13, 15, ARITHMETIC OPERATOR>

<q, 13, 16, IDENTIFIER>  
<\*, 13, 17, ARITHMETIC OPERATOR>  
<sizeof, 13, 18, IDENTIFIER>  
<(, 13, 24, LB>  
<int, 13, 25, KEYWORD>  
<), 13, 28, RB>  
<;, 13, 29, SS>  
<int, 14, 2, KEYWORD>  
<size2, 14, 6, IDENTIFIER>  
<=, 14, 12, ASSIGNMENT OPERATOR>  
<m, 14, 14, IDENTIFIER>  
<\*, 14, 15, ARITHMETIC OPERATOR>  
<n, 14, 16, IDENTIFIER>  
<\*, 14, 17, ARITHMETIC OPERATOR>  
<sizeof, 14, 18, IDENTIFIER>  
<(, 14, 24, LB>  
<int, 14, 25, KEYWORD>  
<), 14, 28, RB>  
<;, 14, 29, SS>  
<int, 15, 2, KEYWORD>  
<arr1, 15, 6, IDENTIFIER>  
<=, 15, 11, ASSIGNMENT OPERATOR>  
<(, 15, 13, LB>  
<int, 15, 14, KEYWORD>  
<), 15, 17, RB>  
<malloc, 15, 18, IDENTIFIER>  
<(, 15, 24, LB>  
<size1, 15, 25, IDENTIFIER>  
<), 15, 30, RB>  
<;, 15, 31, SS>  
<int, 16, 2, KEYWORD>  
<arr2, 16, 6, IDENTIFIER>  
<=, 16, 11, ASSIGNMENT OPERATOR>  
<(, 16, 13, LB>  
<int, 16, 14, KEYWORD>  
<), 16, 17, RB>  
<malloc, 16, 18, IDENTIFIER>  
<(, 16, 24, LB>  
<size2, 16, 25, IDENTIFIER>  
<), 16, 30, RB>  
<;, 16, 31, SS>  
<printf, 17, 2, KEYWORD>  
<(, 17, 8, LB>  
<"Enter the elements of the first array\n", 17, 9, STRING LITERAL>  
<), 17, 48, RB>

<;, 17, 49, SS>  
<int, 18, 2, KEYWORD>  
<i, 18, 6, IDENTIFIER>  
<;, 18, 7, SS>  
<j, 18, 8, IDENTIFIER>  
<;, 18, 9, SS>  
<k, 18, 10, IDENTIFIER>  
<;, 18, 11, SS>  
<for, 19, 2, KEYWORD>  
<(, 19, 6, LB>  
<i, 19, 7, IDENTIFIER>  
<=, 19, 8, ASSIGNMENT OPERATOR>  
<0, 19, 9, NUMBER>  
<;, 19, 10, SS>  
<i, 19, 11, IDENTIFIER>  
<<, 19, 12, RELATIONAL OPERATOR>  
<p, 19, 13, IDENTIFIER>  
<;, 19, 14, SS>  
<i, 19, 15, IDENTIFIER>  
<++, 19, 16, UNARY OPERATOR>  
<), 19, 18, RB>  
<{, 19, 19, LC>  
<for, 20, 3, KEYWORD>  
<(, 20, 6, LB>  
<j, 20, 7, IDENTIFIER>  
<=, 20, 8, ASSIGNMENT OPERATOR>  
<0, 20, 9, NUMBER>  
<;, 20, 10, SS>  
<j, 20, 11, IDENTIFIER>  
<<, 20, 12, RELATIONAL OPERATOR>  
<q, 20, 13, IDENTIFIER>  
<;, 20, 14, SS>  
<j, 20, 15, IDENTIFIER>  
<++, 20, 16, UNARY OPERATOR>  
<), 20, 18, RB>  
<{, 20, 19, LC>  
<scanf, 21, 4, IDENTIFIER>  
<(, 21, 9, LB>  
<"%d", 21, 10, STRING LITERAL>  
<;, 21, 12, SS>  
<arr1, 21, 13, IDENTIFIER>  
<+, 21, 17, ARITHMETIC OPERATOR>  
<q, 21, 18, IDENTIFIER>  
<\*, 21, 19, ARITHMETIC OPERATOR>  
<i, 21, 20, IDENTIFIER>

<+, 21, 21, ARITHMETIC OPERATOR>  
<j, 21, 22, IDENTIFIER>  
<), 21, 23, RB>  
<;, 21, 24, SS>  
<}, 22, 3, RC>  
<}, 23, 2, RC>  
<printf, 24, 2, KEYWORD>  
<(, 24, 8, LB>  
<"Enter the elements of the second array\n", 24, 9, STRING LITERAL>  
<), 24, 49, RB>  
<;, 24, 50, SS>  
<for, 25, 2, KEYWORD>  
<(, 25, 6, LB>  
<i, 25, 7, IDENTIFIER>  
<=, 25, 8, ASSIGNMENT OPERATOR>  
<0, 25, 9, NUMBER>  
<;, 25, 10, SS>  
<i, 25, 11, IDENTIFIER>  
<<, 25, 12, RELATIONAL OPERATOR>  
<m, 25, 13, IDENTIFIER>  
<;, 25, 14, SS>  
<i, 25, 15, IDENTIFIER>  
<++, 25, 16, UNARY OPERATOR>  
<), 25, 18, RB>  
<{, 25, 19, LC>  
<for, 26, 3, KEYWORD>  
<(, 26, 6, LB>  
<j, 26, 7, IDENTIFIER>  
<=, 26, 8, ASSIGNMENT OPERATOR>  
<0, 26, 9, NUMBER>  
<;, 26, 10, SS>  
<j, 26, 11, IDENTIFIER>  
<<, 26, 12, RELATIONAL OPERATOR>  
<n, 26, 13, IDENTIFIER>  
<;, 26, 14, SS>  
<j, 26, 15, IDENTIFIER>  
<++, 26, 16, UNARY OPERATOR>  
<), 26, 18, RB>  
<{, 26, 19, LC>  
<scanf, 27, 4, IDENTIFIER>  
<(, 27, 9, LB>  
<"%d", 27, 10, STRING LITERAL>  
<,, 27, 12, SS>  
<arr2, 27, 13, IDENTIFIER>  
<+, 27, 17, ARITHMETIC OPERATOR>

<n, 27, 18, IDENTIFIER>  
<\*, 27, 19, ARITHMETIC OPERATOR>  
<i, 27, 20, IDENTIFIER>  
<+, 27, 21, ARITHMETIC OPERATOR>  
<j, 27, 22, IDENTIFIER>  
<), 27, 23, RB>  
<;, 27, 24, SS>  
<}, 28, 3, RC>  
<}, 29, 2, RC>  
<int, 30, 2, KEYWORD>  
<size3, 30, 6, IDENTIFIER>  
<=, 30, 12, ASSIGNMENT OPERATOR>  
<p, 30, 14, IDENTIFIER>  
<\*, 30, 15, ARITHMETIC OPERATOR>  
<n, 30, 16, IDENTIFIER>  
<\*, 30, 17, ARITHMETIC OPERATOR>  
<sizeof, 30, 18, IDENTIFIER>  
<(, 30, 24, LB>  
<int, 30, 25, KEYWORD>  
<), 30, 28, RB>  
<;, 30, 29, SS>  
<int, 31, 2, KEYWORD>  
<res, 31, 6, IDENTIFIER>  
<=, 31, 10, ASSIGNMENT OPERATOR>  
<(, 31, 12, LB>  
<int, 31, 13, KEYWORD>  
<), 31, 16, RB>  
<malloc, 31, 17, IDENTIFIER>  
<(, 31, 23, LB>  
<size3, 31, 24, IDENTIFIER>  
<), 31, 29, RB>  
<;, 31, 30, SS>  
<int, 32, 2, KEYWORD>  
<sum, 32, 6, IDENTIFIER>  
<=, 32, 10, ASSIGNMENT OPERATOR>  
<0, 32, 12, NUMBER>  
<;, 32, 13, SS>  
<for, 33, 2, KEYWORD>  
<(, 33, 5, LB>  
<i, 33, 6, IDENTIFIER>  
<=, 33, 7, ASSIGNMENT OPERATOR>  
<0, 33, 8, NUMBER>  
<;, 33, 9, SS>  
<i, 33, 10, IDENTIFIER>  
<<, 33, 11, RELATIONAL OPERATOR>

<p, 33, 12, IDENTIFIER>  
<;, 33, 13, SS>  
<i, 33, 14, IDENTIFIER>  
<++, 33, 15, UNARY OPERATOR>  
<), 33, 17, RB>  
<{, 33, 18, LC>  
<for, 34, 3, KEYWORD>  
<(, 34, 6, LB>  
<j, 34, 7, IDENTIFIER>  
<=, 34, 8, ASSIGNMENT OPERATOR>  
<0, 34, 9, NUMBER>  
<;, 34, 10, SS>  
<j, 34, 11, IDENTIFIER>  
<<, 34, 12, RELATIONAL OPERATOR>  
<n, 34, 13, IDENTIFIER>  
<;, 34, 14, SS>  
<j, 34, 15, IDENTIFIER>  
<++, 34, 16, UNARY OPERATOR>  
<), 34, 18, RB>  
<{, 34, 19, LC>  
<for, 35, 4, KEYWORD>  
<(, 35, 7, LB>  
<k, 35, 8, IDENTIFIER>  
<=, 35, 9, ASSIGNMENT OPERATOR>  
<0, 35, 10, NUMBER>  
<;, 35, 11, SS>  
<k, 35, 12, IDENTIFIER>  
<<, 35, 13, RELATIONAL OPERATOR>  
<m, 35, 14, IDENTIFIER>  
<;, 35, 15, SS>  
<k, 35, 16, IDENTIFIER>  
<++, 35, 17, UNARY OPERATOR>  
<), 35, 19, RB>  
<{, 35, 20, LC>  
<sum, 36, 5, IDENTIFIER>  
<+, 36, 8, ARITHMETIC OPERATOR>  
<=, 36, 9, ASSIGNMENT OPERATOR>  
<arr1, 36, 11, IDENTIFIER>  
<[, 36, 15, LS>  
<i, 36, 16, IDENTIFIER>  
<\*, 36, 17, ARITHMETIC OPERATOR>  
<m, 36, 18, IDENTIFIER>  
<+, 36, 19, ARITHMETIC OPERATOR>  
<k, 36, 20, IDENTIFIER>  
<], 36, 21, RS>

<\*, 36, 23, ARITHMETIC OPERATOR>  
<arr2, 36, 25, IDENTIFIER>  
<[, 36, 29, LS>  
<k, 36, 30, IDENTIFIER>  
<\*, 36, 31, ARITHMETIC OPERATOR>  
<n, 36, 32, IDENTIFIER>  
<+, 36, 33, ARITHMETIC OPERATOR>  
<j, 36, 34, IDENTIFIER>  
<], 36, 35, RS>  
<;, 36, 36, SS>  
<}, 37, 4, RC>  
<res, 38, 4, IDENTIFIER>  
<[, 38, 7, LS>  
<i, 38, 8, IDENTIFIER>  
<\*, 38, 9, ARITHMETIC OPERATOR>  
<n, 38, 10, IDENTIFIER>  
<+, 38, 11, ARITHMETIC OPERATOR>  
<j, 38, 12, IDENTIFIER>  
<], 38, 13, RS>  
<=, 38, 15, ASSIGNMENT OPERATOR>  
<sum, 38, 17, IDENTIFIER>  
<;, 38, 20, SS>  
<sum, 39, 4, IDENTIFIER>  
<=, 39, 7, ASSIGNMENT OPERATOR>  
<0, 39, 8, NUMBER>  
<;, 39, 9, SS>  
<}, 40, 3, RC>  
<}, 41, 2, RC>  
<printf, 42, 2, KEYWORD>  
<(, 42, 8, LB>  
<"The result is\n", 42, 9, STRING LITERAL>  
<), 42, 24, RB>  
<;, 42, 25, SS>  
<for, 43, 2, KEYWORD>  
<(, 43, 5, LB>  
<i, 43, 6, IDENTIFIER>  
<=, 43, 7, ASSIGNMENT OPERATOR>  
<0, 43, 8, NUMBER>  
<;, 43, 9, SS>  
<i, 43, 10, IDENTIFIER>  
<<, 43, 11, RELATIONAL OPERATOR>  
<p, 43, 12, IDENTIFIER>  
<;, 43, 13, SS>  
<i, 43, 14, IDENTIFIER>  
<++, 43, 15, UNARY OPERATOR>



<), 43, 17, RB>  
<{, 43, 18, LC>  
<for, 44, 3, KEYWORD>  
<(, 44, 6, LB>  
<j, 44, 7, IDENTIFIER>  
<=, 44, 8, ASSIGNMENT OPERATOR>  
<0, 44, 9, NUMBER>  
<;, 44, 10, SS>  
<j, 44, 11, IDENTIFIER>  
<<, 44, 12, RELATIONAL OPERATOR>  
<n, 44, 13, IDENTIFIER>  
<;, 44, 14, SS>  
<j, 44, 15, IDENTIFIER>  
<++, 44, 16, UNARY OPERATOR>  
<), 44, 18, RB>  
<{, 44, 19, LC>  
<printf, 45, 4, KEYWORD>  
<(, 45, 10, LB>  
<"%d ", 45, 11, STRING LITERAL>  
<;, 45, 14, SS>  
<\*, 45, 15, ARITHMETIC OPERATOR>  
<(, 45, 16, LB>  
<res, 45, 17, IDENTIFIER>  
<+, 45, 20, ARITHMETIC OPERATOR>  
<n, 45, 21, IDENTIFIER>  
<\*, 45, 22, ARITHMETIC OPERATOR>  
<i, 45, 23, IDENTIFIER>  
<+, 45, 24, ARITHMETIC OPERATOR>  
<j, 45, 25, IDENTIFIER>  
<), 45, 26, RB>  
<), 45, 27, RB>  
<;, 45, 28, SS>  
<}, 46, 3, RC>  
<printf, 47, 3, KEYWORD>  
<(, 47, 9, LB>  
<"\n", 47, 10, STRING LITERAL>  
<), 47, 12, RB>  
<;, 47, 13, SS>  
<}, 48, 2, RC>  
<return, 49, 2, KEYWORD>  
<0, 49, 9, NUMBER>  
<;, 49, 10, SS>  
<}, 50, 1, RC>

