```
NAME: BIJAY KHATRI && DEEPAK VERMA
ROLL NO. : 12/CS/45 && 12/CS/46
GROUP No.: 22
ASSIGNMENT : 3

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
int ptr_value= 100;
int token_index=0;
char* delimiters=" #+-*/%<=>!&|={},;()[]\" \'";
char* keywords[]={"if","else","while","for"};
char* arithmetic[]={"+","-"};
char* conditional[]={"<","<=",">",">=","!="};
char* assignment[]={"="};
char* punctuation[]={"(",")"};
char lines[500][100];
int i=0;
typedef struct tok
{

      char t_name[30];
      char t_type[30];
      int token_code;
      int pointer_value;

}tokens;

tokens ctokens[10000];
int searchToken(char *token)
{
      int i;
      for(i=0;i<token_index;i++)
      {
            if(strcmp(ctokens[i].t_name,token)==0)
                  return 1;
      }
      return 0;
}
int compare(const tokens* a, const tokens* b)
{
      if(a->token_code == b->token_code)
            return a->pointer_value > b->pointer_value;
      return a->token_code > b->token_code;
}
int checkOther(char* token,char* ln)
{
      if(strstr(ln,token))
            return 1;

      return 0;
}
int checkKeyword(char* token,int len)
{
      int temp_index;
      char **ptr= keywords;

      for(temp_index=0;temp_index<len;temp_index++)
            if(strcmp(token,ptr[temp_index])==0)
```

```
                return temp_index;

      return -1;
}
void getLine(FILE *fp)
{
      int j=0;
      char ch;
      while(1)
      {
            ch=getchar();
            if(ch==EOF)
                  break;
            if(ch=='\'' || ch=='\"')
            {
                  while((ch=getchar())!='\n')
                        ;
            }

            if(ch=='\n')
            {
                  i++,j=0;
            }
            else
            {
                  lines[i][j]=ch;
                  j++;
            }
      }
}
char *strstrip(char *s)
{
    size_t size;
    char *end;

    size = strlen(s);

    if (!size)
      return s;

    end = s + size - 1;
    while (end >= s && isspace(*end))
      end--;
    *(end + 1) = '\0';

    while (*s && isspace(*s))
      s++;

    return s;
}
void lex()
{
      int k,len;
      for(k=0;k<=i;k++)
      {
            char ls[100];
            strcpy(ls,lines[k]);
            char* str= ls;

            str= strstrip(str);
```

```
            char* pch = strtok (str,delimiters);

            while (pch != NULL)
              {

                  len=sizeof(keywords)/sizeof(keywords[0]);

                  int tcode=0;

                  if(searchToken(pch)==0 && (tcode=checkKeyword(pch,len))!=-1)
                  {

                        strcpy(ctokens[token_index].t_name,pch);
                        strcpy(ctokens[token_index].t_type,"KEYWORD");
                        ctokens[token_index].token_code = tcode+1;
                        ctokens[token_index].pointer_value=0;
                        token_index++;
                  }

                  else
                        {

                              pch = strstrip(pch);

                              int flag=1;
                              if(strchr(pch,'.') || strcmp(pch,"include")==0)
                                    flag=0;

                              char *ptr= pch;
                              int isNumber=1;

                              while(*ptr!='\0')
                              {
                                    if(!isdigit(*ptr))
                                    {
                                          isNumber=0;
                                          break;
                                    }
                                    ptr++;
                              }

                              if(isNumber)
                              {
                                    strcpy(ctokens[token_index].t_name,pch);

      strcpy(ctokens[token_index].t_type,"CONSTANT");
                                    ctokens[token_index].token_code = 6;
                                    ctokens[token_index].pointer_value=
ptr_value;
                                    ptr_value++;
                                    token_index++;
                              }
                              else if(flag)
                              {
                                    if(searchToken(pch)==0)
                                    {

      strcpy(ctokens[token_index].t_name,pch);
```

```
      strcpy(ctokens[token_index].t_type,"IDENTIFIER");
                                          ctokens[token_index].token_code = 5;
                                          ctokens[token_index].pointer_value=
ptr_value;
                                          ptr_value++;
                                          token_index++;
                                    }
                              }
                        }
                  pch = strtok (NULL, delimiters);
            }
           int tmp;
           for(tmp=0; tmp<sizeof(conditional)/sizeof(conditional[0]);tmp++)
           {
                 if(searchToken(conditional[tmp])==0)
                 {
                       if(strstr(lines[k],conditional[tmp]))
                       {
                             int pos= strstr(lines[k],conditional[tmp])-
lines[k];
                             if((strlen(conditional[tmp])==1) && lines[k]
[pos+1]=='=')
                             {
                                   continue;
                             }
                       strcpy(ctokens[token_index].t_name,conditional[tmp]);
                       strcpy(ctokens[token_index].t_type,"CONDITIONAL");
                       ctokens[token_index].token_code = 7;
                       ctokens[token_index].pointer_value= tmp+1;
                       token_index++;
                       }
                 }
           }
           for(tmp=0; tmp<sizeof( arithmetic)/sizeof(arithmetic[0]);tmp++)
           {
                 if(searchToken(arithmetic[tmp])==0 &&
strstr(lines[k],arithmetic[tmp]))
                 {
                       strcpy(ctokens[token_index].t_name,arithmetic[tmp]);
                       strcpy(ctokens[token_index].t_type,"ARITHMETIC");
                       ctokens[token_index].token_code = 9;
                       ctokens[token_index].pointer_value= tmp+1;
                       token_index++;
                 }
           }

           for(tmp=0; tmp<sizeof(assignment)/sizeof(assignment[0]);tmp++)
           {
                 if(searchToken(assignment[tmp])==0)
                 {
                       if(strstr(lines[k],assignment[tmp]))
                       {
                       strcpy(ctokens[token_index].t_name,assignment[tmp]);
                       strcpy(ctokens[token_index].t_type,"ASSIGNMENT");
                       ctokens[token_index].token_code = 10;
                       ctokens[token_index].pointer_value=0;
                       token_index++;
                       }
                 }
           }
```

```
            for(tmp=0; tmp<sizeof(punctuation)/sizeof(punctuation[0]);tmp++)
            {
                  if(searchToken(punctuation[tmp])==0 &&
strstr(lines[k],punctuation[tmp]))
                  {
                        strcpy(ctokens[token_index].t_name,punctuation[tmp]);
                        strcpy(ctokens[token_index].t_type,"PUNCTUATION");
                        ctokens[token_index].token_code = 8;
                        ctokens[token_index].pointer_value= tmp+1;
                        token_index++;
                  }
            }

      }
}
int main()
{
      FILE *fp;
      fp= fopen("newtest.c","r");
      getLine(fp);
     lex();
     qsort(ctokens,token_index,sizeof(tokens),*compare);
     int nm=0;
     printf("\n%4s%12s\t|\t%10s\t|%13s|\n","TOKEN_CODE","TOKEN","TYPE","POINTER
VALUE");

printf("_____________________________________________________________________\n");
      for(;nm<token_index;nm++)
      {
            if(ctokens[nm].pointer_value==0)
                  printf("%4d\t|%12s\t|\t%10s\t|
%11s|\n",ctokens[nm].token_code,ctokens[nm].t_name,ctokens[nm].t_type,"-");
            else
            printf("%4d\t|%12s\t|\t%10s\t|
%11d|\n",ctokens[nm].token_code,ctokens[nm].t_name,ctokens[nm].t_type,ctokens[nm
].pointer_value);
      }
}

INPUT :

int main()
{     int t,i,j=5;
      /* code */
      scanf("%d",&t);
      for (int i = 0; i < t; ++i)
      {
            /* code */
            cout<<i<<endl;

      }
      while(t>0)
      {
            if(t%2==0)
                  printf("EVEN\n");
            else
                  printf("ODD\n");
            t--;
      }
      return 0;
```

```
}
```

OUTPUT :

TOKEN_CODE	TOKEN	TYPE	POINTER VALUE
5	iostream	IDENTIFIER	100
5	return	IDENTIFIER	129
2	else	KEYWORD	-
5	cstdio	IDENTIFIER	101
5	cstdlib	IDENTIFIER	102
5	using	IDENTIFIER	103
5	namespace	IDENTIFIER	104
5	std	IDENTIFIER	105
5	int	IDENTIFIER	106
5	a	IDENTIFIER	107
6	10	CONSTANT	108
6	10	CONSTANT	109
5	b	IDENTIFIER	110
6	100	CONSTANT	111
6	2	CONSTANT	112
5	i	IDENTIFIER	113
5	j	IDENTIFIER	114
5	k	IDENTIFIER	115
5	count	IDENTIFIER	116
5	main	IDENTIFIER	117
5	printf	IDENTIFIER	128
6	0	CONSTANT	130
5	t	IDENTIFIER	118
6	5	CONSTANT	119
6	0	CONSTANT	127
5	code	IDENTIFIER	120
5	scanf	IDENTIFIER	121
4	for	KEYWORD	-
6	0	CONSTANT	122
6	2	CONSTANT	126
5	cout	IDENTIFIER	123
5	endl	IDENTIFIER	124
3	while	KEYWORD	-
6	0	CONSTANT	125
1	if	KEYWORD	-
7	<	CONDITIONAL	1
7	>	CONDITIONAL	3
8	(	PUNCTUATION	1
8	)	PUNCTUATION	2
9	+	ARITHMETIC	1
9	-	ARITHMETIC	2
10	=	ASSIGNMENT	-