Aditya Pradhan Cse D 180905350

Q1)
Code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <stdbool.h>
int rowcount=1,colcount=0;
typedef struct{
	char token_name[20];
	int rno,cno;
}token;
void createToken(char name[])
{
	token t;
	strcpy(name,token_name);
	t.rno=rowcount;
	t.cno=colcount;
	dispToken(t);
}
void dispToken(token t)
{
	printf("<%s,%d,%d>\n",t.token_name,t.rno,t.cno);
}
void ignoreStringLiteral(FILE *fa)
{
	int ca=getc(fa);
	while(ca!=" \"")
	{if(ca=='\n')
	{
		rowcount++;
		colcount=0;
	}
	else colcount++;
	ca=getc(fa);
	}
}
void ignoreComments(FILE *fa)
{
	int ca;
	int cb=getc(fa);
		if(cb=='/')		// //...
		{
		do{
		ca=getc(fa);
		}while(ca!='\n');
		}
		else if(cb=='*')
		do{
			do{
```

Aditya Pradhan Cse D 180905350

```
                                ca=getc(fa);
                        }while(ca!='*');
                        ca=getc(fa);
                }while(ca!='/');
}
int iskeyword(char buf[])
{
        char keyw[25][10]={"if","else","for","break","switch","do","while",
        "return","void","int","float","char","struct","define","include",
         "\0"};
         for(int j=0;strcmp(keyw[j],"\0")!=0;j++)
                        {
                                if(strcmp(buf,keyw[j])==0)
                                        printf("%s is a keyword\n",(keyw[j]));
                        }
}
bool isArithOperator(char ch)
{
   if (ch == '+' || ch == '-' || ch == '*' ||
      ch == '/' || ch == '%' || ch == '=')
      return (true);
   return (false);
}
bool isSpecialSymbol(char ch)
{
        if(ch=='(' || ch==')' || ch=='{' || ch=='}' || ch=='[' || ch==']' || ch==';')
        return true;
        else return false;
}
bool isRelationalOp(char a,char b)
{
        char str[]=
        if(a=='<')
        {
                if(b=='=')createToken()
        }
}
bool isLogicalOp(FILE *fa)
{
        int ca,cb;

}

char keywords[][100]={"int","float","double",
                                "if","else","switch",
                                "void","char","for",
                                "while","do","break",
                                "continue","case","default",
                                "printf","scanf","return",
                                "bool","true","false",
```

Aditya Pradhan Cse D 180905350

```
                                        "short","long"};
char datatype[][100]={"short","long","int","float","double","void","char","bool"};


struct node
{
        char lexeme[20];
        int size;
        char type[20];
        char scope;
};
typedef struct node symbol;

struct TOKEN
{
        char token_name[100];
        int index;
        unsigned int row,col;
        char type[100];
};

symbol table[100][100];
char func[100][100];
int pres[100]={0};
typedef struct TOKEN token;
FILE *fa;
int row=1,col=1,fnum=-1;
char c,buff2[100];
int dton=0;

void insert(char buffer[])
{

        symbol sy;
        for(int i=0;i<pres[fnum];i++)
        {
                if(strcmp(table[fnum][i].lexeme,buffer)==0)
                        return;
        }

        strcpy(sy.lexeme,buffer);
        strcpy(sy.type,buff2);
        if(strcmp(buff2,"Func")==0)
                sy.size=-1;
        else
        {
                int size=1,trace=0,val=0;
                if(c=='[')
                {
                        c=getc(fa);
```

```
                        while(isdigit(c))
                        {
                                trace++;
                                val=val*10+(c-'0');
                                c=getc(fa);
                        }
                        size=val;
                        fseek(fa,-1*trace,SEEK_CUR);
                }
                if(strcmp(buff2,"int")==0)
                        sy.size=size*4;
                else if(strcmp(buff2,"short")==0)
                        sy.size=size*2;
                else if(strcmp(buff2,"long")==0)
                        sy.size=size*8;
                else if(strcmp(buff2,"char")==0)
                        sy.size=size*1;
                else if(strcmp(buff2,"float")==0)
                        sy.size=size*4;
                else if(strcmp(buff2,"double")==0)
                        sy.size=size*8;
                else if(strcmp(buff2,"bool")==0)
                        sy.size=size*1;
        }
        table[fnum][pres[fnum]]=sy;
        pres[fnum]++;
}

token getNextToken()
{
        token t;int i=0,j=0;

        if(c=='\n')
        {
                row++;
                col=1;
                c=getc(fa);
                printf("\n");
        }
        if(c==' ')
        {
                col++;
                c=getc(fa);
        }
        if(c=="")
        {
                dton=0;
                strcpy(t.token_name,"string");
                t.row=row;
                t.col=col;
```

```
		col++;
		c=getc(fa);
		while(c!="")
		{
			col++;
			c=getc(fa);
		}
		c=getc(fa);
		return t;
	}
	else if(isdigit(c))
	{
		dton=0;
		strcpy(t.token_name,"num");
		t.row=row;
		t.col=col;
		while(isdigit(c))
		{
			col++;
			c=getc(fa);
		}
		return t;
	}
	else if(isalpha(c))
	{
		i=0;
		t.row=row;
		t.col=col;
		char buffer[100];
		while(isalpha(c))
		{
			buffer[i++]=c;
			c=getc(fa);
			col++;
		}
		buffer[i]='\0';
		int z=0;
		for(j=0;j<23;j++)
		{
			if(strcmp(keywords[j],buffer)==0)
			{

				z=1;
				break;
			}
		}
		if(z==1)
		{
			strcpy(t.token_name,buffer);
			for(j=0;j<8;j++)
```

```
                {
                        if(strcmp(datatype[j],buffer)==0)
                        {
                                z=0;
                                break;
                        }
                }
                if(z==0){
                        strcpy(buff2,buffer);
                        dton=1;
                }
        }

        else
        {
                strcpy(t.token_name,"id");
                if(dton==1 && c=='(')
                {
                        fnum++;
                        strcpy(func[fnum],buffer);

                }
                else if(dton==0 && c=='(')
                {
                        strcpy(buff2,"Func");
                        insert(buffer);
                }
                else
                {
                        insert(buffer);
                }
        }
        return t;
}
else if(c=='=')
{
        dton=0;
        t.row=row;
        t.col=col;
        col++;
        c=getc(fa);
        if(c=='=')
        {
                strcpy(t.token_name,"==");
                col++;
                c=getc(fa);
        }
        else
        {
                strcpy(t.token_name,"=");
```

```
		}
		return t;
	}
	else if(c=='+'||c=='-'||c=='*'||c=='/'||c=='%'||c=='<'||c=='>'||c=='!')
	{
		dton=0;
		t.row=row;
		t.col=col;
		i=0;
		char buff[100];
		buff[i++]=c;
		c=getc(fa);
		col++;
		if(c=='=')
		{
			buff[i++]=c;
			c=getc(fa);
			col++;
		}
		buff[i]='\0';
		strcpy(t.token_name,buff);
		return t;
	}
	else if(c=='&')
	{
		dton=0;
		t.row=row;
		t.col=col;
		c=getc(fa);
		col++;
		if(c=='&')
		{
			strcpy(t.token_name,"&&");
		}
		else
		{
			strcpy(t.token_name,"&");
		}
		return t;
	}
	else if(c=='|')
	{
		dton=0;
		t.row=row;
		t.col=col;
		c=getc(fa);
		col++;
		if(c=='|')
		{
			strcpy(t.token_name,"|");
```

```
                }
                else
                {
                        strcpy(t.token_name,"|");
                }
                return t;
        }
        else if(c!='\n'){
        dton=0;
        t.row=row;
        t.col=col;
        char buffer[100];
        buffer[0]=c;
        buffer[1]='\0';
        strcpy(t.token_name,buffer);
        col++;
        c=getc(fa);
        return t;
        }
        else
        {
                strcpy(t.token_name,"\n");
                return t;
        }

}

int main()
{

        FILE *fb;
        int ca,cb,i,j,k;
        char bb[10000];
        fa=fopen("input.c","r");
        if(fa==NULL)
        {
                printf("Cannot open file\n");
                exit(0);
        }
        ca = getc(fa);
        while(ca!=EOF)
        {

                if(ca=='\n' || ca==' ')
                {
                        if(ca=='\n'){
                                rowcount++;
                                colcount=0;
                        }
                        else colcount++;
```

```
                continue;
        }
        if(ca=="")
        {
                ignoreStringLiteral(fa);

        }

        if(ca=='#' )
        {
                while(ca!='\n')
                        ca=getc(fa);
                ca=getc(fa);
                continue;
        }

        if(ca=='/' )
                ignoreComments(FILE *fa);

        else{
                strcpy(ca,bb);
        }
        ca=getc(fa);
}

int i=0;
c=bb[0];
while(c!='\0')
{
        int k;
        token t=getNextToken();
        while(strcmp(t.token_name,"\n")==0)
        {
                t=getNextToken();
        }
        printf("<%s,%d,%d>",t.token_name,t.row,t.col);
        i++;
        c=bb[i];
}
for(i=0;i<=fnum;i++)
{
        printf("\n\nFunction name : %s\n",func[i]);
        printf("\t\tLexeme\tType\tSize\n");
        for(j=0;j<pres[i];j++)
        {
                printf("%d\t\t%s\t%s\t%d\n",j+1,table[i][j].lexeme,table[i][j].type,table[i]
[j].size);
        }
}
return 0;
```

Aditya Pradhan Cse D 180905350

}

Input

```
student@dslab:~/180905350/cd/lab4$ cat input.c
int sum(int a, int b)
{
        int s=a+b;
        return s;
}

bool search(int *arr,int key)
{
        int i;
        for(i=0;i<10;i++){
                if(arr[i]==key)
                        return true;
                else return false;
        }
}

void main()
{
        int a[20],i,sum;
        bool status;
        printf("Enter array elements:");

        for(i=0;i<10;++i)
                scanf("%d",&a[i]);

        sum=a[0]+a[4];
        status=search(a,sum);
        printf("%d",status);
student@dslab:~/180905350/cd/lab4$
```

Aditya Pradhan Cse D 180905350

Ouput

```
student@dslab:~/180905350/cd/lab4$ cc p.c
student@dslab:~/180905350/cd/lab4$ ./a.out
<int,1,1><id,1,5><(,1,8><int,1,9><id,1,13><,,1,14><int,1,16><id,1,20><),1,21>
<{,2,1>
<int,3,2><id,3,6><=,3,7><id,3,8><+,3,9><id,3,10><;,3,11>
<return,4,2><id,4,9><;,4,10>
<},5,1>

<bool,7,1><id,7,6><(,7,12><int,7,13><*,7,17><id,7,18><,,7,21><int,7,22><id,7,26><),7,29>
<{,8,1>
<int,9,2><id,9,6><;,9,7>
<for,10,2><(,10,5><id,10,6><=,10,7><num,10,8><;,10,9><id,10,10><<,10,11><num,10,12><;,10,14><id,10,15><+,10,16><+,10,17><),10,18><{,10,19>
<if,11,2><(,11,4><id,11,5><[,11,8><id,11,9><],11,10><==,11,11><id,11,13><),11,16>
<return,12,2><true,12,9><;,12,13>
<else,13,2><return,13,7><false,13,14><;,13,19>
<},14,2>
<},15,1>

<void,17,1><id,17,6><(,17,10><),17,11>
<{,18,1>
<int,19,2><id,19,6><],19,7><num,19,8><],19,9><,,19,10><id,19,11><,,19,12><id,19,13><;,19,16>
<bool,20,2><id,20,7><;,20,13>
<printf,21,2><(,21,8><string,21,9><),21,31><;,21,32>

<for,23,2><(,23,5><id,23,6><=,23,7><num,23,8><;,23,9><id,23,10><<,23,11><num,23,12><;,23,14><+,23,15><+,23,16><id,23,17><),23,18>
<scanf,24,2><(,24,7><string,24,8><,,24,11><&,24,12><id,24,13><[,24,14><id,24,15><],24,16><),24,17><;,24,18>

<id,26,2><=,26,5><id,26,6><[,26,7><num,26,8><],26,9><+,26,10><id,26,11><[,26,12><num,26,13><],26,14><;,26,15>
<id,27,2><=,27,8><id,27,9><(,27,15><id,27,16><,,27,17><id,27,18><),27,21><;,27,22>
<printf,28,2><(,28,8><string,28,9><,,28,12><id,28,13><),28,19><;,28,20>
<},29,1>

Function name : sum
                Lexeme  Type    Size
1               a       int     4
2               b       int     4
```

```
Function name : sum
                Lexeme  Type    Size
1               a       int     4
2               b       int     4
3               s       int     4


Function name : search
                Lexeme  Type    Size
1               arr     int     4
2               key     int     4
3               i       int     4


Function name : main
                Lexeme  Type    Size
1               a       int     80
2               i       int     4
3               sum     int     4
4               status  bool    1
5               search  Func    -1
student@dslab:~/180905350/cd/lab4$
```