CD Lab 4

Aniruddha Amit Dutta
180905488

Roll no – 58

Q1. **Symbol table**

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

int TableLength = 0;
#define SZ 20

struct token
{
  char lexeme[SZ];
  int idx;
  unsigned int row,col; //row number, column number.
  char type[SZ];
  int sz;
};

struct ListElement{
  struct token tok;
  struct ListElement *next;
};

struct ListElement *TABLE[SZ];

int row,col;
char ca,temp[20];
bool FILE_NOT_ENDED = true;

void print_token(struct token s){
   printf("<%s,%d,%d>",s.lexeme,s.row,s.col);
   return;
}

bool is_include(char* temp){
        if(strstr(temp,"#include")!=NULL){
                        return true;
                }
        return false;
}
```

```
bool is_define(char *temp){
	if(strstr(temp,"#define")!=NULL){
			return true;
		}
	return false;
}



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


int isKeyword(char* word){
	// printf(" word in func %s\n",word );
	for(int i = 0; i < 35; i++){
		// printf(" key in func %s\n",key[i] );
		if(strcmp(key[i], word) == 0) {
			// printf("%s\n", "Keyworh hai ");
			return 1;
		}
		// printf(" strcmp %d\n",strcmp(key[i], word)  );
	}
	return 0;
}

char * datatype[] = {
	"int","float","char","bool"

};

int is_datatype(char * dbuff){
	for(int i = 0; i < 4; i++){
		if(strcmp(datatype[i], dbuff) == 0) {
			return 1;
		}
	}
	return 0;
}

// refactored

bool isDelimiter(char ch)
{
   if (ch == ' ' || ch == ',' || ch == ';'  || ch=='='||
       ch == '(' || ch == ')' || ch == '[' || ch == ']' || ch == '{' || ch == '}')
       return true;
```

```
    return false;
}

bool isRelational_operator(char ch)
{
    if (ch == '>' || ch == '<' || ch == '!')
        return true;
    return false;
}

bool isArithmetic_operator(char ch)
{
    if (ch == '%' || ch == '+' || ch == '-' || ch == '*' ||
        ch == '/' )
        return true;
    return false;
}


bool isRealNumber(char* str)
{
    int i, len = strlen(str);
    bool hasDecimal = false;

    if (len == 0)
        return (false);
    for (i = 0; i < len; i++) {
        if (str[i] != '0' && str[i] != '1' && str[i] != '2'
            && str[i] != '3' && str[i] != '4' && str[i] != '5'
            && str[i] != '6' && str[i] != '7' && str[i] != '8'
            && str[i] != '9' && str[i] != '.' ||
            (str[i] == '-' && i > 0))
            return (false);
        if (str[i] == '.')
            hasDecimal = true;
    }
    return hasDecimal;
}


bool isInteger(char* str)
{
    int i, len = strlen(str);

    if (len == 0)
        return (false);
    for (i = 0; i < len; i++) {
        if (str[i] != '0' && str[i] != '1' && str[i] != '2'
            && str[i] != '3' && str[i] != '4' && str[i] != '5'
            && str[i] != '6' && str[i] != '7' && str[i] != '8'
            && str[i] != '9'|| str[i]=='.' || (str[i] == '-' && i > 0))
            return (false);
```

```
    }
    return (true);
}

int val=-1;

int SEARCH(struct token tk){

          if(val<0)return 0;
    for(int i=0;i<=val;i++){
           struct ListElement * cur = TABLE[i];
           while(cur){
      if(strcmp((cur->tok).lexeme,tk.lexeme)==0&&strcmp((cur->tok).type,tk.type)==0&&(cur-
>tok).idx==tk.idx){
           return 1;
      }
      cur=cur->next;
           }
   }
          return 0;
}

void INSERT(struct token tk){
  if(SEARCH(tk)==1){
     return;
   }

  if(strcmp(tk.type,"func")==0){
              val++;
  }
  struct ListElement* cur = malloc(sizeof(struct ListElement));
  cur->tok = tk;
  cur->next = NULL;
   (cur->tok).idx=val;

  if(TABLE[val]==NULL){
     TABLE[val] = cur; // No collosion.
  }
  else{
    struct ListElement * ele= TABLE[val];
    while(ele->next!=NULL){
       ele = ele->next; // Add the element at the End in the case of a collision.
    }
    ele->next = cur;
  }
}



char buff[40],dbuff[40];
int mul=0;

struct token getNextToken(FILE *fa){
```

```
        char cb;
        char word[20], num[20];
        int i = 0;
        num[0]='\0';
while(ca != EOF){
    struct token s;
                if(ca == '\n'){
                        row++;
                        col = 1;
                        printf("\n");
                }
    else if(ca=='#'){
                        int x=0;
                        while(ca!='\n'){
                                temp[x++] = ca;
                                ca = getc(fa);
                                col++;
                        }
                        temp[x] = '\0';
                        if(!(is_define(temp)||is_include(temp))){
                                strcpy((s.lexeme),temp);
                                s.row=row;
                                s.col=col-strlen(temp);
                                strcpy(s.type,"unknown");
                                s.sz=sizeof(temp);
                                col=1;
          return s;
                        }
                        col=1;
                }
                // remove comments , blankspaces
                else if(ca==' '||ca=='\t'){
        ca=fgetc(fa);
                        while(ca==' '||ca=='\t'){
                                ca=fgetc(fa);
                        }
        fseek(fa,-1,SEEK_CUR);

                }
                else if (ca=='/'){
                        cb = getc(fa);
                        if (cb == '/'){
                                while(ca != '\n')
                                        ca = getc(fa);
                                col=0;
                        }
                        else if (cb == '*'){
                                do{
                                        while(ca != '*')
                                                ca = getc(fa);
                                        ca = getc(fa);
                                }while (ca != '/');;
```

```
        }
        else{
                fseek(fa, -2, SEEK_CUR);
        }
}
// check string
else if(ca == ""){
        // printf(" 5 \n");
        strcpy(s.lexeme,"string literal");
        s.row=row;
        s.col=col;
        print_token(s);
        ca = getc(fa);
        while(ca != ""){
                col++;
                ca = getc(fa);
        }
        col++;
}

// is a word ->  keyword / variable
else if(isalpha(ca)) {
        i=0;
        while(isalpha(ca) || isdigit(ca) || ca == '_'){
                word[i++] = ca;
                ca = getc(fa);
                col++;
        }
        word[i]='\0';
        fseek(fa,-1,SEEK_CUR);
        col--;
        //printf(" word = %s\n",word );
        if(isKeyword(word)){
                        strcpy(s.lexeme,word);
                        strcpy(buff,word);
                        s.row=row;
                        s.col=col-(int)(strlen(word))+1;
                        return s;
   }
        else{
                        if(is_datatype(word)){
                                strcpy(dbuff,word);
                        }
                        // printf("iddd\n");
                    char name[20]="";
                    strcat(name,"id ");
                    strcat(name,word);
                    strcpy(s.lexeme,name);
                    ca=fgetc(fa);

                    // reset
                    if(strcmp(buff,"func")==0)
```

```
                                        buff[0]='\0';

                                    // check if variable name or function name
                                        if(ca=='(')
                                                strcpy(buff,"func");
                                    fseek(fa,-1,SEEK_CUR);

                                    strcpy(s.type,buff);
                                        s.row=row;
                                        s.col=col-(int)(strlen(word))+1;
                                        if(strcmp(buff,"int")==0)
                                                s.sz=sizeof(int);
                                        else if(strcmp(buff,"char")==0)
                                                s.sz=sizeof(char);
                                        else if(strcmp(buff,"bool")==0)
                                                s.sz=sizeof(bool);
                                        else if(strcmp(buff,"func")==0)
                                                s.sz=-1;
                                        // for int a[20]
                                        else if( ( ca=fgetc(fa) ) == '[' ){
                                                // printf("here ptr = %c",ca);
                                                ca=fgetc(fa);
                                                int h=0;
                                                while( isdigit( ca )  ){
                                                        mul = mul*10;
                                                        mul += (ca)-'0';
                                                        // printf("here mul = %d",mul);
                                                        ca=fgetc(fa);
                                                }
                                                mul =0;
                                                s.sz = mul*(s.sz);
                                                // fseek(fa,-1,SEEK_CUR);
                                        }
                                        else{
                                                fseek(fa,-1,SEEK_CUR);
                                                s.sz=0;
                                        }
                                        //printf("bef\n");
                                        if(strcmp(buff,"return")==0||strcmp(buff,"if")==0||
strcmp(buff,"scanf")==0||strcmp(buff,"printf")==0||strcmp(buff,"for")==0)
                                                return s;
                                        INSERT(s);
                                        //printf("after\n");
                                        //buff[0]='\0';
                                        return s;
                        }

                }

                // is an Delimeter
                else if(isDelimiter(ca)){
                                char  c[10];
```

```
                                        c[0]=ca;
                                        c[1]='\0';
                                        strcpy(s.lexeme,c);
                                        s.row=row;
                                        s.col=col;
                                        col++;
                                        return s;
                }
// is a relational op
else if(isRelational_operator(ca)){
        char  c[10];
                                        c[0]=ca;
                                        c[1]='\0';
                                        strcpy(s.lexeme,c);
                                        ca=getc(fa);
                                        col++;
                                        s.row=row;
                                        if(ca=='=')
                                                s.col=col-1;
                                        else{
                                                s.col=col;
                                                fseek(fa,-1,SEEK_CUR);
                                        }
                                        return s;
}
else if(isArithmetic_operator(ca)){
        char  c[10];
                                        c[0]=ca;
                                        c[1]='\0';
                                        strcpy(s.lexeme,c);
                                        s.col=col;
                                        s.row=row;
                                        return s;

}
            // is a number of any sort
            else if(isdigit(ca)){
                    i=0;
                    num[i++] = ca;
                    while(isdigit(ca)|| ca == '.'){
                            num[i++] = ca;
                            ca = getc(fa);
                            col++;
                    }
                    num[i]='\0';
                    if(isRealNumber(num) || isInteger(num)){
                                        strcpy(s.lexeme,"num");
                                        s.row=row;
                                        s.col=col- (int)(strlen(num))+1;

                                        return s;
                }
```

```
                        i = 0;
                        num[0]='\0';
                        continue;
                }
                //col++;
                ca = getc(fa);
                //end of while
        }
        FILE_NOT_ENDED = false;
        struct token s;
        strcpy(s.lexeme,"null");
        strcpy(s.type,"null");
        s.row=-1;
        s.col=-1;
        return s;
}

void Initialize(){
  for(int i=0;i<SZ;i++){
     TABLE[i] = NULL;
  }
}

void Display(){
 for(int i=0;i<=val;i++){
        struct ListElement * cur = TABLE[i];
    printf("%d %s %s\n\n",i+1,(cur->tok).lexeme,(cur->tok).type);
        cur=cur->next;
        while(cur){
                printf("%s   %s   %d\n",(cur->tok).lexeme, (cur->tok).type,(cur->tok).sz);
                cur=cur->next;
        }
    printf("********************\n");
 }
}

int main(int argc, char const *argv[])
{
        FILE *fa=fopen("input.txt","r");
        struct token s;
        row=1;
        col=1;
        ca=fgetc(fa);
        Initialize();
        while(FILE_NOT_ENDED&&ca!=EOF){
                s=getNextToken(fa);
                ca=fgetc(fa);
       print_token(s);
       // printf("after call to gettoken +1 ca is = %c",ca);
        }
        printf("\nSYMBOL TABLE\n");
        Display();
```

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

//

```
File  Edit  View  Search  Terminal  Help
$ gcc token.c
$ ./a.out
<int,1,1><id sum,1,3><(,1,5><int,1,6><id a,1,8><,,1,8><int,1,9><id b,1,11><),1,11>
<{,2,1><int,2,2><id s,2,4><=,2,4><id a,2,5><+,2,5><id b,2,5><;,2,5>
<return,3,1><id s,3,6><;,3,6>
<},4,1>
<bool,5,1><id search,5,4><(,5,9><int,5,10><*,5,12><id arr,5,12><,,5,14><int,5,15><id key,5,17><),5,19>
<num,6,1>
<int,7,1><id i,7,3><;,7,3>
<for,8,1><(,8,3><id i,8,4><=,8,4><num,8,5><id i,8,6><<,8,7><num,8,7><id i,8,9><+,8,9><+,8,9><),8,9><{,8,10>
<if,9,1><(,9,2><id arr,9,3><],9,5><=,9,6><=,9,7><id key,9,8><),9,10>
<return,10,1><id true,10,6><;,10,9>
<else,11,1><return,11,4><id false,11,9><;,11,13>
<},12,1>
<},13,1>
<void,14,1><id main,14,4><(,14,7><),14,8>
<{,15,1>
<int,16,1><id a,16,3><[,16,3><num,16,4><,,16,6><id i,16,7><,,16,7><id sum,16,8><;,16,10>
<bool,17,1><id status,17,4><;,17,9>
<printf,18,1><(,18,6><string literal,18,7><),18,29><;,18,30>
<for,19,1><(,19,3><id i,19,4><=,19,4><num,19,5><id i,19,6><<,19,7><num,19,7><+,19,9><+,19,9><id i,19,9><),19,9>
<scanf,20,1><(,20,5><string literal,20,6><,,20,9><id a,20,10><],20,10><),20,11><;,20,12>
<id sum,21,1><=,21,3><id a,21,4><+,21,4><id a,21,4><;,21,4>
<id status,22,1><=,22,6><id search,22,7><(,22,12><id a,22,13><,,22,13><id sum,22,14><),22,16><;,22,17>
<printf,23,1><(,23,6><string literal,23,7><,,23,10><id status,23,11><),23,16><;,23,17>
<},24,1>
SYMBOL TABLE
1 id sum func

id a   int   4
id b   int   4
id s   int   4
******************
2 id search func
```

```
<id sum,21,1><=,21,3><id a,21,4><+,21,4><id a,21,4><;,21,4>
<id status,22,1><=,22,6><id search,22,7><(,22,12><id a,22,13><,,22,13><id sum,22,14><),22
<printf,23,1><(,23,6><string literal,23,7><,,23,10><id status,23,11><),23,16><;,23,17>
<},24,1>
SYMBOL TABLE
1 id sum func

id a   int   4
id b   int   4
id s   int   4
********************
2 id search func

id arr   int   4
id key   int   4
id i   int   4
********************
3 id main func

id i   int   4
id sum   int   4
id status   bool   1
********************
4 id search func

id a      0
id sum      0
********************
$
```

```
3 id main func

id i   int   4
id sum   int   4
id status   bool   1
********************
4 id search func

id a      0
id sum      0
********************
$ cat input.txt
int sum(int a, int b)
{ int s=a+b;
return s;
}
bool search(int *arr,int key)
25{
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);
```