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

CD LAB - 7 : RD PARSER FOR DECLARATION STATEMENTS

LAB EXERCISES :

P1) For given subset of grammar 7.1, design RD parser with appropriate error messages with expected character and row and column number.

Program → main () { declarations assign_stat }
declarations→ data-type identifier-list; declarations | ∈
data-type → int | char
identifier-list → id | id, identifier-list
assign_stat → id=id; | id = num;

Program :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
struct token
{
  char lexeme[64];
  int row,col;
  char type[20];
};
static int row=1,col=1;
char buf[2048];

const char specialsymbols[]={'?',';',':',','};
const char *keywords[] = {"const", "char", "int","return","for", "while",
"do",
            "switch", "if", "else","unsigned", "case", "break" };

const char *datypes[]={"int","char","void","float","bool"};
```

```
const char arithmeticsymbols[]={'*'};
int isdtype(char *w)
{
  int i;
  for(i=0;i<sizeof(datypes)/sizeof(char*);i++)
  {
    if(strcmp(w,datypes[i])==0)
    {
      return 1;
    }}
  return 0;
}
int isKeyword(const char *str)
{
  for(int i=0;i<sizeof(keywords)/sizeof(char*);i++)
     {
      if(strcmp(str,keywords[i])==0)
        {
          return 1;
        }
     }
  return 0;
}

int charBelongsTo(int c,const char *arr)
{
  int len;
  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 *tkn,char c,int row,int col, char *type)
{
  tkn->row=row;
  tkn->col=col;
  strcpy(tkn->type,type);
  tkn->lexeme[0]=c;
  tkn->lexeme[1]='\0';
}

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

struct token getNextToken(FILE *f1)
{
  int c;
  struct token tkn=
  {
    .row=-1
  };
  int gotToken=0;
  while(!gotToken && (c=fgetc(f1))!=EOF)
     {
      if(charBelongsTo(c,specialsymbols))
        {
          fillToken(&tkn,c,row,col,"specialsymbol");
          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,"left bracket");
     gotToken=1;
     ++col;
    }
else if(c==')')
    {
     fillToken(&tkn,c,row,col,"right bracket");
     gotToken=1;
     ++col;
    }
else if(c=='{')
    {
     fillToken(&tkn,c,row,col,"left curly");
     gotToken=1;
     ++col;
    }
else if(c=='}')
    {
     fillToken(&tkn,c,row,col,"right curly");
     gotToken=1;
     ++col;
    }
  else if(c=='+')
    {
     int d=fgetc(f1);
     if(d!='+')
       {
        fillToken(&tkn,c,row,col,"arithmetic operator");
        gotToken=1;
        ++col;
        fseek(f1,-1,SEEK_CUR);
       }
```

```
      else
        {
          fillToken(&tkn,c,row,col,"unary operator");
          strcpy(tkn.lexeme,"++");
          gotToken=1;
          col+=2;
        }
      }
else if(c=='-')
      {
       int d=fgetc(f1);
       if(d!='-')
        {
          fillToken(&tkn,c,row,col,"arithmetic operator");
          gotToken=1;
          ++col;
          fseek(f1,-1,SEEK_CUR);
        }
       else
        {
          fillToken(&tkn,c,row,col,"unary operator");
          strcpy(tkn.lexeme,"--");
          gotToken=1;
          col+=2;
        }
      }
else if(c=='=')
      {
       int d=fgetc(f1);
       if(d!='=')
        {
          fillToken(&tkn,c,row,col,"assignment operator");
          gotToken=1;
          ++col;
          fseek(f1,-1,SEEK_CUR);
        }
       else
        {
          fillToken(&tkn,c,row,col,"relational operator");
```

```
            strcpy(tkn.lexeme,"==");
            gotToken=1;
            col+=2;
           }
        }
else if(isdigit(c))
     {
      tkn.row=row;
      tkn.col=col++;
      tkn.lexeme[0]=c;
      int k=1;
      while((c=fgetc(f1))!=EOF && isdigit(c))
         {
          tkn.lexeme[k++]=c;
          col++;
         }
      tkn.lexeme[k]='\0';
      strcpy(tkn.type,"number");
      gotToken=1;
      fseek(f1,-1,SEEK_CUR);
     }
else if(c == '#')
     {
      while((c = fgetc(f1)) != EOF && c != '\n');
      newLine();
     }
else if(c=='\n')
     {
      newLine();
      c = fgetc(f1);
    if(c == '#')
      {
      while((c = fgetc(f1)) != EOF && c != '\n');
      newLine();
        }
      else if(c != EOF)
            {
        fseek(f1, -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(f1))!= 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(f1,-1,SEEK_CUR);
    }
else if(c=='/')
    {
     int d=fgetc(f1);
     ++col;
     if(d=='/')
       {
         while((c=fgetc(f1))!= EOF && c!='\n')
            {
             ++col;
            }
         if(c=='\n')
```

```
        {
          newLine();
        }
      }
    else if(d=='*')
        {
         do
          {
          if(d=='\n')
            {
              newLine();
            }
          while((c==fgetc(f1))!= EOF && c!='*')
             {
              ++col;
              if(c=='\n')
                {
                  newLine();
                }
             }
           ++col;
          }while((d==fgetc(f1))!= EOF && d!='/' && (++col));
         ++col;
        }
    else
      {
       fillToken(&tkn,c,row,--col,"arithmetic operator");
       gotToken=1;
       fseek(f1,-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(f1)) != 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(f1);
    if(d == '=')
      {
      ++col;
      strcat(tkn.lexeme, "=");
      }
    else
      {
      if(c == '!')
        {
        strcpy(tkn.type, "logical operator");
        }
      fseek(f1, -1, SEEK_CUR);
      }
    gotToken = 1;
     }
else if(c == '&' || c == '|')
    {
    int d = fgetc(f1);
    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
        {
        fseek(f1, -1, SEEK_CUR);
        }
       ++col;
       }
   else
     {
     ++col;
     }
 }
 return tkn;
}


void program();
void declarations();
void data_type();
void identifier_list();
void assign_stat();

struct token curr;
FILE *f1;
void invalid(){
        printf("error");
        exit(0);
}
void program()
{

        if(strcmp(curr.lexeme,"main")==0)
        {
                curr=getNextToken(f1);
                if(strcmp(curr.lexeme,"(")==0)
                {
                        curr=getNextToken(f1);
                        if(strcmp(curr.lexeme,")")==0)
                        {
```

```
                        curr=getNextToken(f1);
                        if(strcmp(curr.lexeme,"{")==0)
                        {
                                curr=getNextToken(f1);
                                declarations();
                                assign_stat();
                                if(strcmp(curr.lexeme,"}")==0)
                                {
                                        return;
                                }
                                else
                        {
                                printf("\nMissing } at row:%d and col:%d.\n\
n",curr.row,curr.col);
                                exit(0);
                        }
                        }
                        else
                        {
                                printf("\nMissing { at row:%d and col:%d.\n\
n",curr.row,curr.col);
                                exit(0);
                        }
                }
                else
                        {
                                printf("\nMissing ) at row:%d and col:%d.\n\
n",curr.row,curr.col);
                                exit(0);
                        }
        }
        else
                        {
                                printf("\nMissing ( at row:%d and col:%d.\n\
n",curr.row,curr.col);
                                exit(0);
                        }
    }
    else
```

```
                        {
                                printf("\nMissing main function\n\n");
                                exit(0);
                        }
}

void declarations()
{

        if(isdtype(curr.lexeme)==0)
        {
                return;
        }
        data_type();
        identifier_list();
        if(strcmp(curr.lexeme,";")==0)
        {
                curr=getNextToken(f1);
                declarations();
        }
        else {printf("\nMissing ; at row:%d and col:%d.\n\
n",curr.row,curr.col); exit(0);}
}

void data_type()
{
        if(strcmp(curr.lexeme,"int")==0)
        {
                curr=getNextToken(f1);
                return;
        }
        else if(strcmp(curr.lexeme,"char")==0)
        {
                curr=getNextToken(f1);
                return;
        }
        else
                        {
```

```
                                printf("\nMissing data type at row:%d and col:
%d.\n\n",curr.row,curr.col);
                                exit(0);
                        }
}

void identifier_list()
{
        if(strcmp(curr.type,"identifier")==0)
        {
                curr=getNextToken(f1);
                if(strcmp(curr.lexeme,",")==0)
                {
                        curr=getNextToken(f1);
                        identifier_list();
                }
                else return;
        }

        else
                {
                        printf("\nMissing identifier at row:%d and col:%d.\n\
n",curr.row,curr.col);
                        exit(0);
                }
}

void assign_stat()
{
        if(strcmp(curr.type,"identifier")==0)
        {
                curr=getNextToken(f1);
                if(strcmp(curr.lexeme,"=")==0)
                {
                        curr=getNextToken(f1);
                        if(strcmp(curr.type,"identifier")==0)
                        {
                                curr=getNextToken(f1);
                                if(strcmp(curr.lexeme,";")==0)
```

```
                        {
                                curr=getNextToken(f1);
                                return;
                        }
                }

        else if(strcmp(curr.type,"number")==0)
        {
                curr=getNextToken(f1);
                        if(strcmp(curr.lexeme,";")==0)
                        {
                                curr=getNextToken(f1);
                                return;
                        }
                        else
                        {
                                printf("\nMissing ; at row:%d and col:%d.\n\
n",curr.row,curr.col);
                                exit(0);
                        }

        }
        else
                        {
                                printf("\nMissing identifier at row:%d and col:
%d.\n\n",curr.row,curr.col);
                                exit(0);
                        }
        }
        else
                        {
                                printf("\nMissing = at row:%d and col:%d.\n\
n",curr.row,curr.col);
                                exit(0);
                        }
    }
    else
                        {
```

```
                                printf("\nMissing identifier at row:%d and col:
%d.\n\n",curr.row,curr.col);
                                exit(0);
                        }
}

int main()
{
        FILE *fa, *fb;
        int ca, cb;
        fa = fopen("input.c", "r");
        if (fa == NULL){
                printf("Cannot open file \n");
                return 0;
        }
        fb = fopen("fileout.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);
fa = fopen("fileout.c", "r");
if(fa == NULL){
        printf("Cannot open file");
        return 0;
}
fb = fopen("temp.c", "w");
ca = getc(fa);
while(ca != EOF){
        if(ca == '#'){
            while(ca != '\n'){
                ca = getc(fa);
            }
        }
        ca = getc(fa);
        if(ca != EOF && ca != '#'){
            putc(ca, fb);
        }
}
fclose(fa);
fclose(fb);
fa = fopen("temp.c", "r");
fb = fopen("fileout.c", "w");
ca = getc(fa);
while(ca != EOF){
        putc(ca, fb);
        ca = getc(fa);
}
```

```
        fclose(fa);
        fclose(fb);
        remove("temp.c");
        f1=fopen("fileout.c","r");
        if(f1==NULL)
          {
                printf("Error! File cannot be opened!\n");
                return 0;
          }
        struct token tkn;
        curr=getNextToken(f1);
        program();
        printf("\nCompiled\n\n");
    fclose(f1);
}
```

Output :

```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/
lab 7$ cat input.c
#include <stdio.h>
main()
{
        int x,y,z;
        char a;
        x = 9;
}rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB
/lab 7$gcc lab7_p1.c -o p1
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/
lab 7$ ./p1

Compiled

rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/
lab 7$
```

```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/
lab 7$ gcc lab7_p1.c -o p1
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/
lab 7$ cat input.c
#include <stdio.h>
main()
{
        int x,y,z;

        x = 99
}rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB
/lab 7$ ./p1

Missing ; at row:6 and col:1.

rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/
lab 7$
```

```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab 7$ cat input.c
#include <stdio.h>
main()
{

}rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab 7$gcc lab7_p1.c -o p1
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab 7$ ./p1

Missing identifier at row:4 and col:1.

rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab 7$ cat input.c
#include <stdio.h>
main()
{
        int x,y,z;
        char a;
        x = 9
}rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab 7$ ./p1

Missing ; at row:6 and col:1.

rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab 7$ cat input.c
#include <stdio.h>
main()
{
        int x;
}rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab 7$./p1

Missing identifier at row:4 and col:1.

rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab 7$ cat input.c
#include <stdio.h>
main()
{
        int x;
        x = 99;
}rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab 7$./p1

Compiled

rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CD_LAB/lab 7$
```