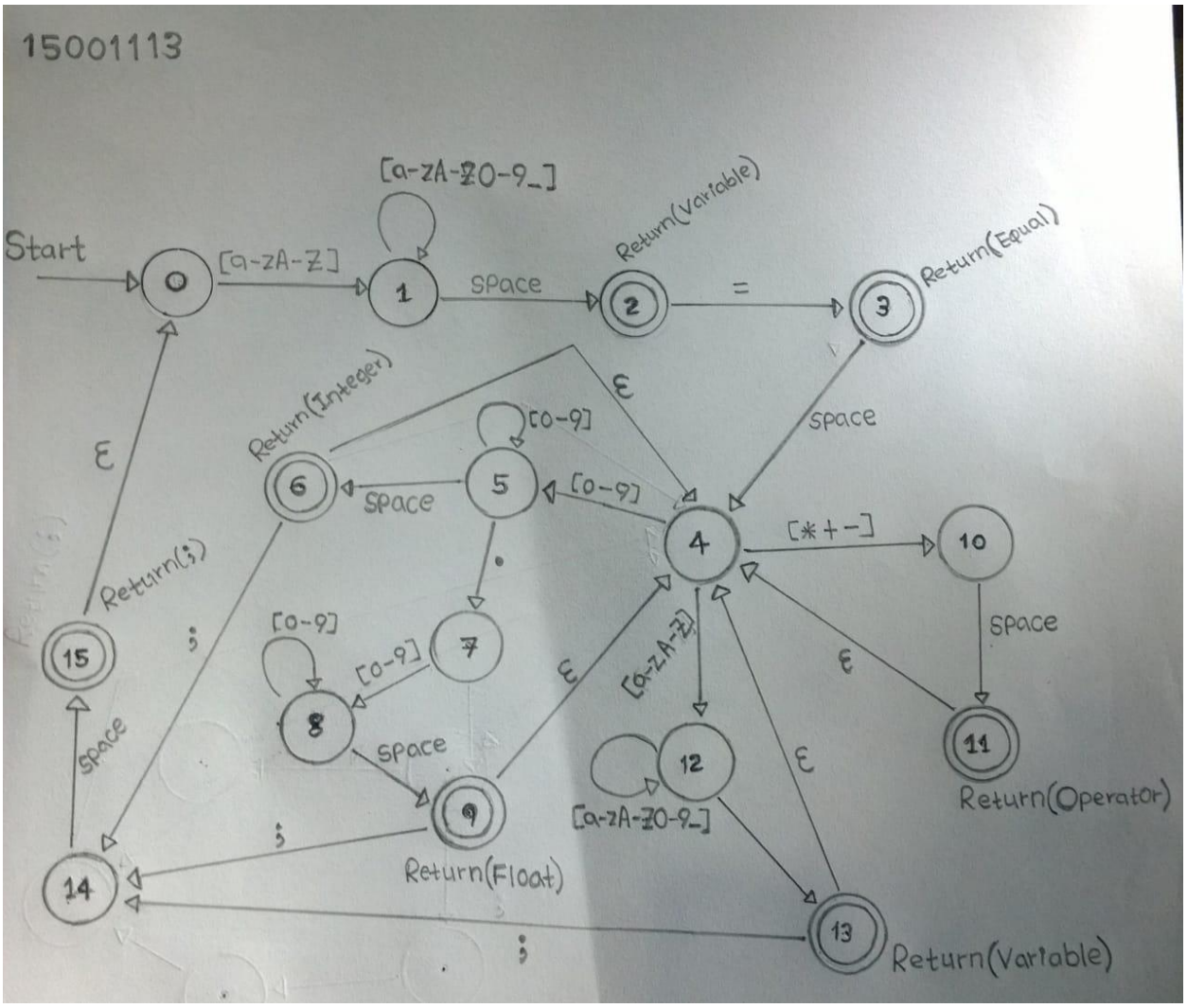
15001113 Yasas Ranawaka 2015CS111 Assignment Lexer

a)*



b)* Here is the github link of code that I uploaded.

https://github.com/yasas1/Lexer-Assignment/blob/master/Tokenize1/main.c

```
#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

int main()

{
```

```
FILE *f;
char c;
char lexeme[20]="\0";
int lexeme_pos = 0;
int state = 0;
int error = 0;
int semi=0; /*semicolon checker.There can't be a semicolon after '=' or the operators */
int lineNo=1; /*to count the new line character(line no)*/

printf("Lineno 1 :");

f=fopen("C:\\Text\\input.txt","r"); // change the location of file
while((!error) && ((c=fgetc(f))!=EOF)){

    lexeme[lexeme_pos++] = c;

    switch(state)
    {
        case 0 :
            if(isalpha(c)){state = 1;}
            else if(c=='\n')
            {
                lineNo++;
                printf("\nLineno %d :",lineNo);
```

```
    }
    else
    {
        printf("Error Detected %c,Firstly there should a variable ",c);
        error = 1;
    }
    break;

case 1 :
    if(isalpha(c) || isdigit(c) || c=='_' ){state = 1;}
    else if(c=='\n')
    {
        lineNo++;
        printf("\nLineno %d :",lineNo);
    }
    else
    {
        if(c == ' ')
        {
            lexeme[--lexeme_pos]='\0'; //remove space
            printf("<%s,%s>","variable",lexeme);
            state = 2;
            lexeme_pos = 0;
        }else
```

```
            {
                printf(" Error Detected %c,,There should be a space character after
the variable",c);
                error = 1;
            }
        }
        break;


    case 2 :


        if(c=='='){state=3;}
        else if(c=='\n')
        {
            lineNo++;
            printf("\nLineno %d :",lineNo);
        }
        else
        {
            printf(" Error Detected %c,,There should be a = sign ",c);
            error = 1;
        }
        break;


    case 3 :
```

```
        if(c==' ')
        {
            lexeme[--lexeme_pos]='\0'; //remove space
            printf("<%s,%s>","Equal",lexeme);
            state = 4;
            lexeme_pos = 0;
            semi=0;
        }
        else if(c=='\n')
        {
            lineNo++;
            printf("\nLineno %d :",lineNo);
        }
        else
        {
            printf(" Error Detected %c,,There should a space character after the =
",c);
            error = 1;
        }
        break;

    case 4 :

        if(isdigit(c)){state = 5;}
        else if(c=='*' || c=='+' || c=='-'){state=10;}
```

```
    else if(isalpha(c)){state=12;}
    else if(c==';')
    {
        state=15;
    }
    else if(c!=' ')
    {
        printf(" Error Detected_%c_, Undefined character",c);
        error = 1;
    }
    else if(c=='\n')
    {
        lineNo++;
        printf("\nLineno %d :",lineNo);
    }
    break;

case 5 :
    if(isdigit(c)){state = 5;}
    else if(c=='\n')
    {
        lineNo++;
        printf("\nLineno %d :",lineNo);
    }
```

```
        else
        {
            if(c == ' ')
            {
                lexeme[--lexeme_pos]='\0';
                printf("<%s,%s>","Integer",lexeme);
                state = 4;
                lexeme_pos = 0;
                semi=1;
            }
            else
            {
                if(c == '.'){state = 7;}
                else
                {
                    printf("Error Detected %c,,There should a space character after
the integer",c);
                    error = 1;
                }

            }
        }
        break;

    case 7 :
```

```
    if(isdigit(c)){state = 8;}
    else if(c=='\n')
    {
        lineNo++;
        printf("\nLineno %d :",lineNo);
    }
    else
    {
        printf(" Error Detected %c,,There should be at least a number ",c);
        error = 1;
    }
    break;

case 8 :

    if(isdigit(c)){state = 8;}
    else if(c=='\n')
    {
        lineNo++;
        printf("\nLineno %d :",lineNo);
    }
    else
    {
```

```
            if(c == ' ')
            {
                lexeme[--lexeme_pos]='\0';
                printf("<%s,%s>","Float",lexeme);
                state = 4;
                lexeme_pos = 0;
                semi=1;
            }
            else
            {
                printf("Error Detected %c,,There should a space character after the
float",c);
                error = 1;
            }
        }
        break;

    case 10 :

        if(c==' ')
        {
            lexeme[--lexeme_pos]='\0'; //remove space
            printf("<%s,%s>","Operator",lexeme);
            state = 4;
            lexeme_pos = 0;
```

```
            semi=0;
        }
        else if(c=='\n')
        {
            lineNo++;
            printf("\nLineno %d :",lineNo);
        }
        else
        {
            printf(" Error Detected %c,,There should a space character after the
Operator ",c);
            error = 1;
        }

        break;

      case 12 :

        if(isalpha(c) || isdigit(c) || c=='_' ){state = 12;}
        else if(c=='\n')
        {
            lineNo++;
            printf("\nLineno %d :",lineNo);
        }
        else
```

```
        {
            if(c == ' ')
            {
                lexeme[--lexeme_pos]='\0'; //remove space
                printf("<%s,%s>","variable",lexeme);
                state = 4;
                lexeme_pos = 0;
                semi=1;
            }else
            {
                printf(" Error Detected %c,,There should be a space character after
the variable",c);
                error = 1;
            }
        }
        break;

    case 15 :

        if(semi==0)
        {
            printf(" Error Detected..There should not be semicolon");
            error = 1;
        }
        else if(c=='\n')
```

```
            {
               lineNo++;
               printf("\nLineno %d :",lineNo);
            }
            else if(c==' ')
            {
               lexeme[--lexeme_pos]='\0'; //remove space
               printf("<%s,%s>","endOfAssignment",lexeme);
               state = 0;
               lexeme_pos = 0;
            }
            else
            {
               printf(" Error Detected %c,,There should a space character after the
Semicolon ",c);
               error = 1;
            }

            break;
      }
   }

   if(error!=1){
```

```
    if(lexeme[0]==';' && state==15){printf("There should be a space character after the semicolon");}

    if(state==4){printf(" Statement should be finish with a semicolon and a space character");}

    if(state==5){printf(" Statement can't be finish with a integer");}

    if(state==8){printf(" Statement can't be finish with a float");}

    if(state==12){printf(" Statement can't be finish with a variable");}

    if(state==10){printf(" Statement can't be finish with a operator");}

  }

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

When you run the code change the file path**.**

Here is the github link of code that I uploaded.

https://github.com/yasas1/Lexer-Assignment/blob/master/Tokenize1/main.c

Assumption

01* each element is separated by a single space character. Even after the semicolon there should be a space character.

02* A variable can be assigned to a variable. (x = y ; true)

03* A new line character can be anywhere while including assumption 01 and 02.

04* I considered '=' sign as a "Equal" token class for easy to implementation.

05* I take first line of text file as "Lineno 1". Then second will be "Lineno 2".

06* In left hand side, there can be only one variable. Because it is said in a question.

07* There can be also underscore (_) for a variable name in middle or end.

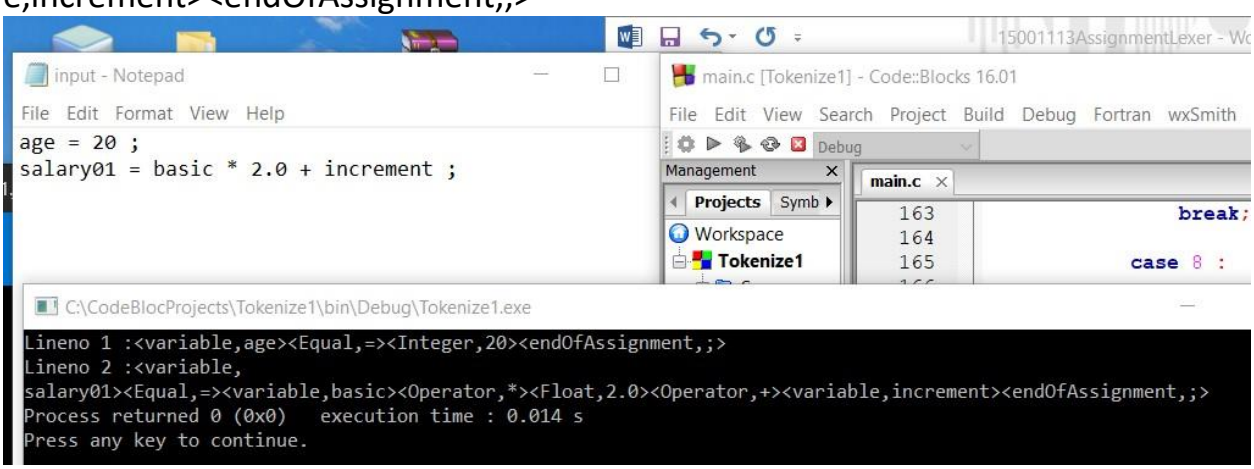
Test Data

01* Fully Correct Inputs

age = 20 ;
salary01 = basic * 2.0 + increment ;
(there should be a space character after the each element. Even after the semicolon ;)

Output

Lineno 1 :<variable,age><Equal,=><Integer,20><endOfAssignment,;>
Lineno 2 :<variable,
salary01><Equal,=><variable,basic><Operator,*><Float,2.0><Operator,+><variable,increment><endOfAssignment,;>



(Identify Text, Integers, Float, Operator, Semicolon and Line no Correctly and Return Token text correctly**)**

In this example, there are also test data for text variable, integer, float , operators, endOfAssignment(;). I will include test data for those again in separately. Because it was asked separately.

02* For Integer: **age = 20 ;**

03* For Float: **age = 20.5 ;**

04* For Operators: **age = 20 + 10 * 1.5 ;**

05* For Text (Variable): salary01 = **sal_basic + 20 * bonus ;**

(Also after the semicolon, there is a space character.)

06* Incorrect Data

a)*Incorrect variable inputs: **1age = 20 ;** (Variable name should be start with a character)

Output

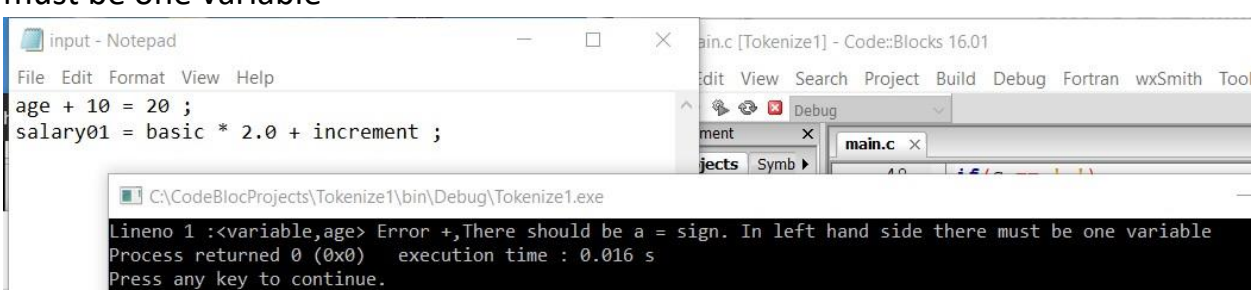
Lineno 1 :Error Detected 1,Firstly there should a variable begin with a character

b)*When In left hand side, there are more elements more than one variables

age + 10 = 20 ;

Output

Lineno 1 :<variable,age> Error +,There should be a = sign. In left hand side there must be one variable



c)* There is a semicolon after a operator (+,-,*) or after = : age = 10 + ;

Output

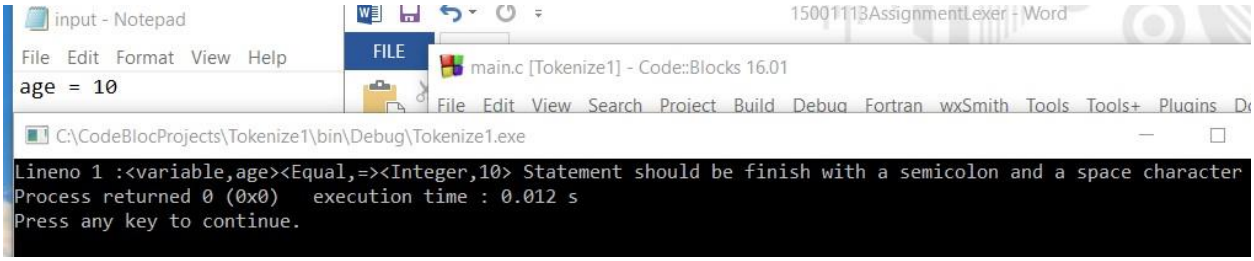
Lineno 1 :<variable,age><Equal,=><Integer,10><Operator,+> Error Detected..There should not be semicolon

d)* When a statement is finish without a semicolon: **age = 10 +**

age = 10

Output

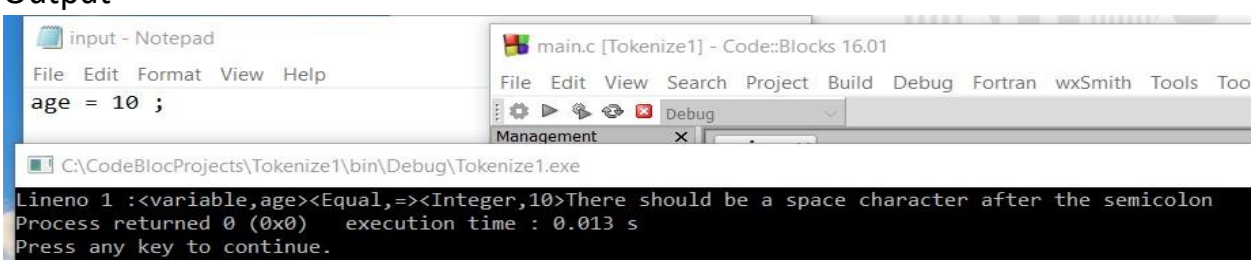
Lineno 1 :<variable,age><Equal,=><Integer,10> Statement should be finish with a semicolon and a space character



e)*When there is not a space character after the semicolon(endOfAssignment)

age = 10 ;

Output



f)* When there is no a space character after any element

age = 20 ;

salary01 = basic* 2.0 + increment;

Output