CSE 351
Programming Languages
Term Project (Spring 2023)

In your term project, you will design and implement a translator that accepts simple C programs, and turns them into pseudocodes. Here is an example:

C

```
#include <stdio.h>

void main(void)
{
    int a,b=3,d=f,Factorial=1;
    char str='d',str2;

    int i;
    int MAX;
    scanf ("%d",&MAX);
    for (i=1;i<=MAX;i++)
    {
        if(MAX<0){
          Factorial=-1;
          i=MAX+1;
        }
        else if(MAX<10){
          Factorial=Factorial*i;
        }
        else{
          Factorial=0;
          i=MAX+1;
        }


    }
    printf ("%d\n",Factorial);
    a=multFunc(a,b);

}

int multFunc( int x, int y)
{
    x= x*y;
    return x;
}
```

Pseudocode

```
FUNCTION main
  b=3
  d=f
  Factorial=1
  str='d'

  READ MAX
  i=1
  WHILE i<=MAX
     IF MAX < 0 THEN
       Factorial=-1
       i=MAX+1
     ELSEIF MAX < 10 THEN
       Factorial=Factorial*i
     ELSE
       Factorial=0;
       i=MAX+1
     ENDIF
     i=i+1
  END WHILE
  PRINT Factorial
  a=multFunc a b
  END FUNCTION main

FUNCTION multFunc x y
     x = x * y
     multFunc=x
END FUNCTION multFunc
```

TODO LIST:

1) **Recognize functions:**

   Each function in input file will be translated into a **Pseudocode** like below. First you must print reserved keyword "***FUNCTION***". Then, print the name of the function in C followed by the list of input parameters. After this, you print the body of function in an appropriate way into the Pseudocode. Finally you will complete the function definition by printing "***END FUNCTION functionName"*** reserved keyword.

```
FUNCTION functionName inputList

//body of the function

END FUNCTION functionName
```

2) **Call functions:**

   a. printf function: All the details about the format will be ignored (see the example). Use the **PRINT** keyword.

   b. scanf function: All the details about the format will be ignored (see the example). Use the **READ** keyword.

   c. A user defined function will be called by calling its name and its parameters

```
multFunc(a,b); // A C function call
multFunc a b   // Its corresponding pseudocode output
```

3) **Return from functions:**

   A return statement with an argument is translated as follows:

```
return(a);        // A C return statement
functionName=a  // Its corresponding pseudocode output,
                // Here, the functionName will be the
                // the name of the function hosting the
                // return statement.
```

4) **Loops: Only C *for* loops will be recognized as input and structure of the input will be as follows:**

```
for (assignment;comparison;increment)
```

- **assignment can be anything like:**
  - a=b
  - j=k;
- **comparison can be anything like:**
  - a==b
  - a<b
  - a<=b
  - a>b
  - a>=b
  - a!=b
  - a==5
  - 5==a
  - etc.

- **Increment can be**
  - i++;
  - i--;
  - i=i+1;

Note that: Any C variable name should be recognized.

INPUT	OUTPUT

`for(assignment;comparison;increment)`
`{`
`  //body`
`}`
	`assignment`
	`WHILE comparison`
	
	`  //body`
	`increment`
	`END WHILE`

5) **If conditions: if else if else if … else condition should be accepted in your design. Input structure will be:**

INPUT	OUTPUT

`if (comparison){`
`  //body`
`}`
`else if (comparison){`
`  //body`
`}`
`else {`
`  //body`
`}`
	`IF comparison THEN`
	`  //body`
	
	`ELSEIF comparison THEN`
	`  //body`
	
	`ELSE`
	`  body`
	`ENDIF`

Note that: There can be more than one "else if".

Note that: There will be only one comparison in an "if condition". That means you do not have to deal with conditions like"a==b && c<d".

BONUS LIST:

6) **Check undeclared variables in input text:** You should check whether a variable is declared or not. If there is an undeclared variable, you should give appropriate error message. Take consider that C language uses static scoping so you should check declaration of variables in functions separately.

7) **Nested Loop and Nested If statements:** You should accept the conditions that each loop can include loops and if statements. For each nested loop or if statements you should use indentation. Example is in the next page.

INPUT	OUTPUT

`for(i=0 ; i<10 ; i++)`
`{`
`  a=a+i;`
`  k=1;`
`  for(j=1 ; j<10 ; j++)`
`  {`
`  k=k*2;`
`  }`
`  if(k==1)`
`  k=0;`
`}`
	i=0
	WHILE i<10
	a=a+i
	k=1
	j=1
	WHILE j<10
	k=k*2
	j=j+1
	END WHILE
	IF k==1
	k=0
	ENDIF
	i=i+1
	END WHILE