CS125 Section 5 *"Algorithm Development"*

#1 **Roles of Variables**. The purpose of most variables can be described using a small number of roles. Recognizing these roles will help you learn how to read, write and understand more complex code. For each example read the whole example and then determine the role of each variable.

Code Example 1	**Most-recent holder:** The value of a most-recent holder is the latest one of a certain group or simply the latest input value. **Gatherer:** The value of a gatherer accumulates all the values gone through so far. **One-way flag:** A one-way flag has two possible values but cannot get its original value anymore after it has been once changed.
Code Example 2	**Fixed value:** Its value is not changed after initialization. **Stepper:** Stepper goes through a succession of values in some systematic way, predictable succession of values. **Follower:** A follower always gets the old value of another known variable (usually most-recent holder) as its new value. **Temporary**: The value of a temporary is only needed for a very short period.

```
int num; // most-recent holder
int sum = 0; // gatherer
boolean done = false; // ?
while(!done){
   num = TextIO.getlnInt();
   if(num==0)
       done = true;
   sum += num;
}
```

```
int TOTAL = 100; // ?
int i = 0; // ?
int num = 0; // ?
int oldNum; // follower
while(i < TOTAL){
  oldNum = num;
  num = TextIO.getlnInt();
  boolean isBigger = (num > oldNum);// temporary
  if(isBigger)
    TextIO.putln("New number is larger");
  i = i + 1;
}
```

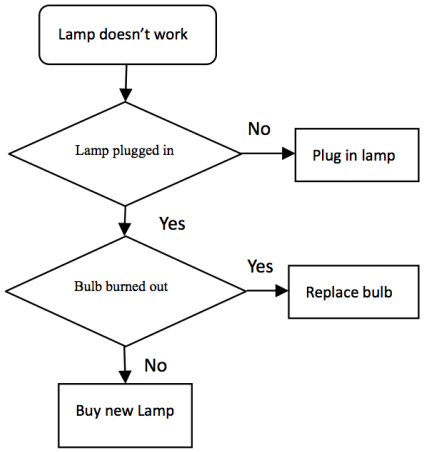
Code Example 3

```
String line = TextIO.getln();
String newString = ""; //?
int i = 0;             //?
char lastChar = ' ';   //?
boolean changedSomething = false; //?
while(i < line.length()) {
  char c = line.charAt(i); //?
  boolean toUC =
      Character.isWhitespace(lastChar)
      && Character.isLetter(c); //?
  if (toUC) {
     c = Character.toUpperCase(c);
     changedSomething = true;
  }
  newString += c;
  lastChar = c;
  i ++;
}
if (changedSomething)
   newString += "!" ;
TextIO.putln(newString);
```

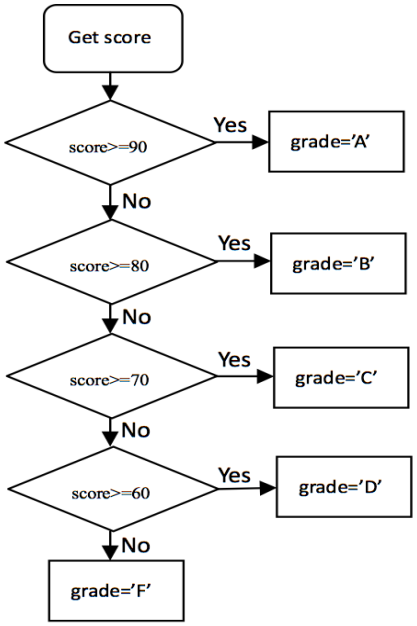
Code Example 4

```
int num;           //?
int expected = 24; //?
boolean found = false; //?
while(!found){
   num = TextIO.getlnInt();
   if(num == expected){
      TextIO.putln("Found number!!");
      found = true;
   }
}
```

#2 Flow charts and if-else structures. Flow charts are a common way to show program flow. Can you convert a flowchart specification into an if-else structure? Study the given the given example then try the scores problem. Carefully check your neighbor's code with scores 90, 80, 70, 60 & 50.




```
// pseudo code
if(lamp plugged in) {
    if(bulb burned out) {
        replace bulb
    }
    else {
        buy new lamp
    }
}
else {
    plug in lamp
}
```




```
public class convertScore {
 public static void main (String args[]) {
  int score = TextIO.getlnInt();
  char grade='?';  // Add code in the space below.









System.out.println("Score: " + score + " Grade: " +
 grade);
 } // end of main method
}  // end of class
```

Concepts that we expect you to know for the first mid-term exam: *Note: This is a representative list that is not necessarily exhaustive.*

Variables and Types: declaration, initialization, assignment, literals, casting, type promotion and type checking;

- boolean, char, int, long, float, double, String

Expressions: order of evaluation, use of parentheses to override order of evaluation, determining the resulting type of an expression

- +, -, *, /, %, &&, ll, ==, <, <=, >, >=, ++, --, and variants w/equal (+=, -=, …)

Conditionals: if, else, using {} to make compound statements, knowing which "if" an "else" goes with.

Loops: for, while, do/while; nested loops, again using { and } for multiple statements

String: length(), charAt(), toUpperCase(), toLowerCase(), equals(), indexOf(), substring(), concatenation (+), indices begin at 0 and end at length-1, converting other types to a string *(e.g.,* "" + (char)'a' + myInt).

Characters: difference with Strings, converting to/from integers, modulo arithmetic.

TextIO: *including:* getln(), getlnChar(), getlnInt(), getlnLong(), getlnDouble(), getlnBoolean(), getlnString(),

getChar(), getInt(), getLong(), getDouble(), getBoolean(), put(), putln(), eof()

Scope: variables can only be used within the scope they are defined.

Syntax: you should know correct Java syntax for the above

Tracing Code: ability to trace code with conditionals, simple arrays and nested loops.

Design: Write short programs that include these elements to solve problems with complexity similar to the MPs.

FSM ideas in MP#2:

In MP#2, you need a very simple FSM for printing out the lines for a particular character in a movie. Many of you probably wrote this code correctly without knowing about FSMs, but this will give you a framework for solving these kind of problems.

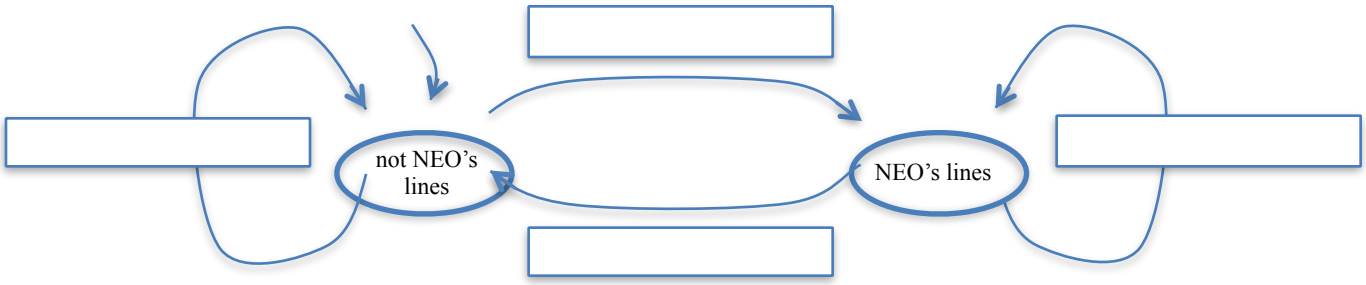
Consider the movie script format:

```
A WOMAN is huddled beside the oven,
peering inside through the cracked
door.

                    NEO
Hello?

              ORACLE   (WOMAN)
I know. You're Neo. Be right with
you.
```

The lines associated with a particular character are written on lines that follow a line that has that character's name in all caps and continue until a blank line is reached. We can build an FSM for printing out NEO's lines with the following: (the FSM does one transition for each line processed.)



This state machine only has two states. As a result, we can maintain the state just as a boolean, like:

```
boolean inNeosLinesState = false; // start state is "not NEO's lines"
```

The only tricks in implementing this code are: 1) correctly transitioning between the states, and 2) only printing out the lines where we were in the "NEO's lines" both last iteration of the loop and this iteration. This can be implemented by correctly ordering the statements that print lines with those that check for state transitions.

Exam Prep

Fix the scoping error:

```
for(int i=0; i<10; i++){ TextIO.putln(i);}
TextIO.putln("i="+i);
```

Loop analysis:

What is the output of the following code?

```
int x=0;
int y = -15;
while(y>2 || y<-2){
  x++;
  y = y / 2 + 1;
  TextIO.putln("x="+x+",y="+y);
  x = 2 * y - x;
}
```

String review:

Modify the following program to print exactly the same output but using *TextIO.put* instead of *TextIO.putln*

```
TextIO.putln("\"Hello");
TextIO.putln("How are you?\"");
```

What is the output of the following code?

```
String str = "CS125 Java Java";
int pos1 = str.indexOf("CS125");
int pos2 = str.indexOf("Java");
for (int i=pos1; i < pos2; i++) {
  System.out.print(str.charAt(i));
}
```

Complete the following program.

Do not create additional variables or print the same letter in two different places in your code. Complete the program below to print out either “C” “D” “F” or “T” (cat, dog, fish or toy respectively). Use the following rules:

A condo dweller gets a cat provided they have $13 or greater to spend.
A house owner gets a dog provided they have more than $17 to spend.
A dorm dweller with $5 or more, and also house owner with exactly $5, gets a fish.
All others should get a stuffed toy.

```
public class PurrfectPetsRecommendation {
   public static void main (String[] args) {
      TextIO.putln("Do you live in a Dorm, Condo or House?");
      TextIO.putln("Type D C or H and press return");
      char type = TextIO.getlnChar();  // assume either D C or H
      TextIO.putln("Dollars you can spend on your pet each week?");
      int dollars = TextIO.getlnInt(); // assume a valid integer









   } // end of main method
}    // end of class
```

Call Log Retrieval

Following text file was recovered from 007's recent call log when his cell phone crashed. Print out the lines containing "Chapman" with the next 2 call logs (following 2 lines), if any. Printed lines should in the original (unmodified) mixture of upper- and lower-case.

"moneYPenny 650-857-0275, James Bond 100-113-0519, 02/14/2012, 08:12AM, 00:01:16
James bOnd 100-113-0519, Chapman 217-001-1407, 02/14/2012, 06:45Am, 00:10:01
M 212-861-9692, JaMes Bond 100-113-0519, 02/13/2012, 10:43pm, 00:00:03
Q 509-553-1081, James bonD 100-113-0519, 02/12/2012, 11:14AM, 00:03:24
JamEs BoND 100-113-0519, SKYlon REStaurant 207-654-7800, 02/10/2012, 04:20pM, 00:01:09
ChApMaN 217-001-1407,James BOnd 100-113-0519, 02/10/2012, 07:12AM, 00:02:22 …"

```
public class CallLogRetrieve {
  public static void main(String [] args) {
    TextIO.readFile("007CallLog.txt")









} } // end of main method and class
```

Advanced: Bond and Chapman may have spoken more than once in any 3 consecutive calls in the call history.