The 2009 Iverson Computing Science Competition
May 26, 2009

(Without Answer Pages)

(Please Print Clearly)

Name: ____

School: ____

City: ____

Grade: _______ **Gender (M/F)** ______________

Computer Teacher ______________________________________

Are you taking AP or IB Computer Science (Yes/No): ______________

Have you completed Programming 5 (Yes/No/Taking it this term): ____________

____

Question	Question Name	Questions to be marked	Assigned Mark
1	FSM - Baby Talk String Checker		
2	Topographical Analysis		
3	The Convict River Crossing Problem		
4	Cyber Sleuth		
5	Namorian Calculator		
	TOTAL		

Exam Format:

This is a traditional 2 hour paper and pencil exam. It consists of five questions with each question consisting of two or more parts. Part one of each question is designed to be relatively easy. The subsequent parts of each question build on the first part adding additional complexity and difficulty.

Students who have just started their work in Computing Science should be able to do reasonably well on part one of each question. More knowledgeable students should be able to do reasonably well on the more difficult parts of each question.

Unlike previous exams there are no required questions. You are to attempt any **four** of the **five** questions doing as much as you can. If you do decide to do more than four questions you **MUST** indicate which four questions you want marked on the title page of the test. Otherwise the markers will mark the first four questions.

Programming Language:

The parts of the questions that require programming can be answered using any programming language you wish (for example, VB, C/C++, Java, or Perl). If you wish you can even use pseudo-code. However, if you do so, be sure to provide an adequate amount of detail so the markers can determine if your solution is correct. **Your pseudo-code should be detailed enough to allow for near direct translation into an appropriate programming language.**

Our primary interest is in your higher order thinking skills rather than on your code wizardry. So a demonstration of logical thinking and systematic problem solving approaches will count for more than a mastery of one particular language's syntax. Be this as it may, you will still have to use some type of coding language to demonstrate what you can do. However minor syntactical errors will be ignored and we encourage you to put in comments where needed to clarify your code.

Suggestions:

1. Read the problem descriptions **carefully**. To get full marks **all** problem specifications for the portions of the questions you attempt have to be met. You can assume that only valid input values will be entered by the user. You do not need to include out-of-range or data type error checks in the input section of your programs.

2. Where feasible, sample executions of the desired program have been included for each problem. Review them carefully to make sure you haven't missed any specifications and to get hints as to how to proceed. In these sample executions the sample data entered by the user are underlined.

3. We **strongly** recommend that you do some design work prior to writing any code. Use pseudo code, diagrams, screen displays, tables or any other aid to help you plan your code. We will be looking at your rough work. If that work is present you won't have to rely completely on your coded solution to get marks. Remember we are looking for the key computing ideas, not specific coding details. In particular you can introduce your own "built-in" functions for sub tasks such as reading the next number, or the next character in a string, or loading an array. Just make sure you specify the pre and post conditions of your "built-in" sub-programs.

4. Take a look at **all** of the questions before deciding on the ones to attempt. Remember if you do decide to do more than four you **MUST** indicate which four questions you want to be marked in the **Questions To Be Marked** column on the first page of the test.

5. Make sure to include English language comments to explain non-obvious or clever parts of your solution.

Question 1: Finite State Machines - A Baby Talk String Checker

A finite-state machine (FSM) is a type of computation device. Although we don't usually encounter them directly, finite-state machines exist inside almost any device that does something interesting: kitchen appliances, remote control toys, simple phones (the complex ones have full computers), video games and so on.

There are three primitive concepts associated with a FSM, each associated with part of a FSM diagram:

State - a circle that indicates the current settings of the parts of the machine. Typically the state has a label that tells you what it means.

Transition - a potential change from one state to another (possibly the same) state. The transition is labeled with the events that cause the transition to occur. There is often a transition named DEFAULT. This is the transition that is taken when the event that occurs does not match a specific transition from the current state.

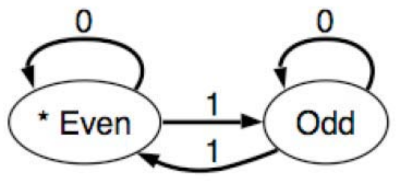
Event -something that causes a transition. For example, pushing a button, or reading a character from the input, or the ticking of a clock are events.

A FSM does a computation by starting in a specific start state (marked with a *), and then waiting for events to occur. Each event causes the machine to change state by following the transitions that matches the event. The computation stops when there are no more events; or an event occurs that does not have a matching transition and there is no DEFAULT transition from the current state.

A good physical intuition for these notions is: A state is a possible location that you can be at on a map. When you are at one location a transition is a road that can take you to another location. An event causes you to choose a particular road to travel. A computation is a road-trip, or path, between a start location and an end location.

So when you trace a FSM computation, think of putting a marker (like a pebble) on the current state, and when an event occurs, moving the marker to the next state along the transition that matches the incoming event.

Example 1: Here is a FSM that processes two possible kinds of events: 0 and 1

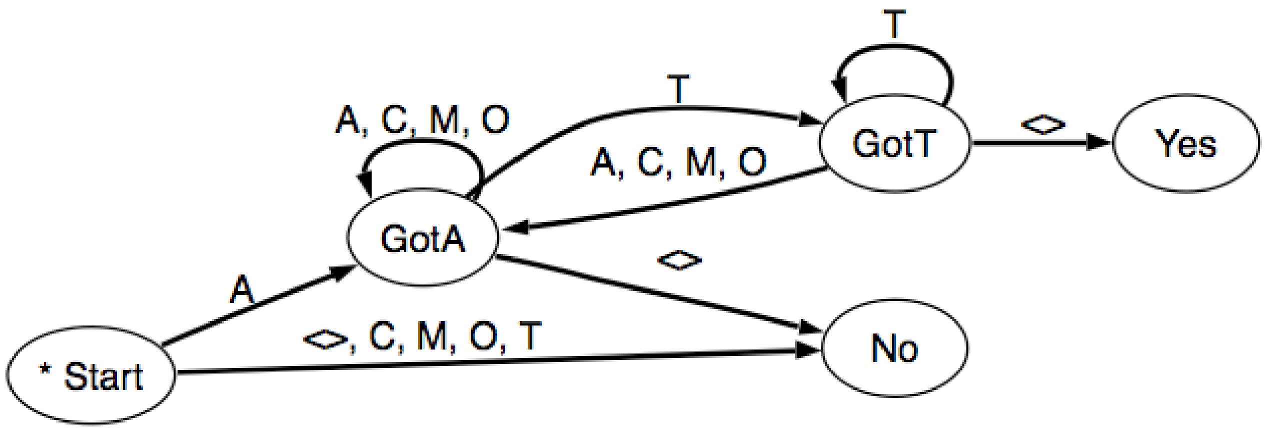


This FSM starts in state **Even** (the * means start here). Every time a 1 event occurs the FSM makes a transition to the other state. But every time a 0 event occurs the FSM makes a transition back to the same state. Thus, the name of the current state tells you whether the number of 1's that have arrived so far is odd (**Odd**) or even (**Even**).

Example 2: One of the most common applications of FSMs is in string pattern matching. The characters of a string come in one by one as events, starting with the left most character of the string. The end of the string is signaled by the special event <>. String pattern matching machines have a starting state called Start; a Yes state that signals if the pattern is found in the string; and a No state if the pattern is not present. Other states are then added as necessary. When the <> event occurs, this means no more events will arrive, so the machine can stop. It can stop earlier if it chooses.

For example, suppose the possible characters are A, C, M, O, T. Here is a machine that looks for strings that start with A and end with T. On the string ATOOT<> this machine makes the following transitions:

Start --A--> GotA --T--> GotT --O --> GotA --O--> GotA --T--> GotT --<>--> Yes



Your Tasks:

A group of scientists have discovered that baby talk isn't just a set of meaningless sounds. It actually is a simple language. While there is still much work to be done, they have discovered that baby talk or rather baby language follows a set of rules.

A baby talk word is:

1. the single letter "A", or
2. the word formed by putting the letter "M" in front of any baby talk word, or
3. the word formed by putting the letter "G" between any two baby talk words.

Here are some examples:

A
AGA
AGAGA
MAGA

The word “A”' is a baby language word because it consists of the single letter "A".
The word “AGA” is a baby language word because it starts with an “A” followed by the letter "G" and the baby language word “A”.
The word “AGAGA” is a baby language word because it starts with an “A” followed by the letter "G" and the baby language word “AGA”.
The word “MAGA” is a baby language word because starts with the letter "M" followed by the baby language word “AGA”.

Part 1. You've been hired by the researchers to design a FSM that will check strings of letters and determine if individual strings are baby language words. Using the diagram on page 4 as a guide, design a FSM that accepts a string of letters and ends in a YES state if the string is a baby language word and in a NO state if it is not.

Strings such as A, AGA, AGAGAGA, MA, MAGAGA and MMA are baby language words while strings such as AG, GAGA and MAGAG are not.

Question 1 - Part 1 - Please draw your diagram here

Part 2. – Advanced Baby Language

Write a program to implement your FSM **but** with the following change.

The scientists did discover one additional rule for baby language. They discovered that babies always pair their M's with an H. So a baby talk word that starts with the letter "M" followed by a baby talk word must end with the letter "H". This can't be properly represented by a finite state machine, but the scientists would like you to incorporate this rule into your program.

Your program should accept a set of possible baby words, determine if they actually are baby language words and print out the results. The output should consist of the string being assessed followed by a YES if the word is a baby language word, and by a NO if the word is not a baby language word. Use the string DONE as a sentinel value to indicate that all of the strings have been processed. Don't display any output for the sentinel value.

In this sample execution the data entered by the user are underlined.

```
Sample Input/Output

Baby Word Checker
  A   YES
  GAGA   NO
  AGAGA   YES
  MA   NO
  MAGMAH   NO
  MMAHH   YES
  DONE
```

Question 1 - Part 2

Question 2: Topographical Analysis

Modern technology allows us to obtain various kinds of geographical data, thus enabling us to construct accurate maps of the earth. With this data we can compute many interesting things, such as the average altitude of an island, volume of water in a lake, size of a land mass, length of a shore, and so on.

Geographical information is typically arranged in a grid of cells, where each cell contains some data associated with the (x,y) coordinate of the cell.

For example, here is a grid map of a lake with an island in it, with altitude measurements relative to the water level taken at intervals of 100 m. A positive number in a cell indicates that on the average the terrain in that cell is above the water level while a negative number indicates that on the average the terrain is below water level. By way of example, the altitude at position (8,4) is -4 or 4 meter below water level while to position at (5,4) is 11 or 11 meters above the water level.

North

Y																
	8	-5	-4	-3	-3	-2	-3	-1	-2	-3	-2	-3	-4	-4	-5	-9
	7	-4	3	4	4	3	3	4	3	-1	2	-2	3	4	-3	-8
	6	-3	2	6	4	3	4	4	2	2	2	1	3	6	4	-7
	5	-4	-2	-1	1	2	2	2	3	-2	2	2	2	2	-1	-6
Y	*4*	-5	1	3	4	8	11	8	-2	-4	-2	3	6	8	4	-5
	3	-2	2	6	12	15	12	6	2	2	3	6	10	12	8	-7
	2	-2	3	8	3	6	5	3	4	2	5	4	6	8	6	-8
	1	-2	-1	3	-1	2	1	1	2	-1	-1	-1	2	6	4	-8
	0	-6	-2	-1	-3	-2	-2	-3	-2	-2	-1	-3	-4	-6	-6	-8
		0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*

X

Your Task: You've been hired by a group of scientists to write one or more programs that transform this type of raw data into more useful information. You can assume that your programs will always be dealing with charts of this type involving a single island completely surrounded by water. The island may or may not have a body of water on it. No cell will ever have a value of '0'. No chart will be larger than 15 by 15.

Important Note: If you decide to do Part 2 of this question you do not have to do Part 1. Part two is really a superset of part 1. However you should be sure that you can do Part 1 before moving on to Part 2

Part 1:
The scientists' first request is for a program that:

1. inputs all data items for a specific row or column of the grid.
2. determines the average altitude of **all** of the points along these coordinates that are above water level
3. determines the highest point along these coordinates and notes the coordinates
4. outputs the information

In this sample execution the data entered by the user are underlined.

Sample Input/Output

```
  Please give X,Y values of the biggest coordinate on the grid.
  Value of X: 14
  Value of Y: 8
  Do you wish to process a row or column (Row or Column): Row
  What Row: 4
  Enter the altitudes for Row 4:
     -5   1   3   4   8   11   8   -2   -4   -2   3   6   8   4   -5

  The average altitude of the area above the water line is 5.6 m
  The highest point is 11 m located at 5,4
```

Question 2 - Part 1

Part 2:
The scientists would like a program that gives them information about the island as a whole. They'd like a program that:

1. inputs all data items for the entire grid
2. determines the average altitude of all of the points that are above water level
3. determines the highest point and notes the coordinates
4. determines the land area of the island in square meters (remember each cell on the grid is 100 meters to a side)
5. determines the number of cells on the shore line, including shore line along lakes inside islands. A cell is considered part of the shore line if it is above the water, and it has at least one cell directly to the north, south, east or west that is below the water.
6. outputs the information

In this sample execution the data entered by the user are underlined.

Sample Input/Output

```
  Please give X,Y values of the biggest coordinate on the grid.
  Value of X: 14
  Value of Y: 8

  Enter the altitudes for Row 0:
     -6   -2   -1   -3   -2   -2   -3   -2   -2   -1   -3   -4   -6   -6   -8
   (Program should prompt user to enter all other rows)

  The average altitude of the area above the water line is 7.25 m
  The highest point is 15 m located at 4,3
  The area of the island is 760000 square meters
  There are 43 cells that have shoreline
```

Question 2 - Part 2

Question 3: The Convict River Crossing Problem

Three peace officers and three convicts were making their way back to their minimum security prison when they came to the bank of a river. The small foot bridge that normally spanned the river was gone, swept away by floods earlier in the spring. However someone had left a small boat tethered to the bank. Unfortunately the boat could only carry two people.

That presented a problem. The convicts were serving time for minor crimes and weren't especially dangerous but they couldn't be allowed to outnumber the peace officers. If that were to happen the convicts would overpower the peace officers, take the key to their restraints and escape. So it is vitally important for your career in the correctional services that there are never more convicts than peace officers on either side of the river.

Note: There is no problem with leaving one or more convicts alone on one of the banks. They know they have no chance of escape unless they can get free of their restraints.

Important Note: If you decide to do Part 2 of this question you do not have to do Part 1. Part two is really a superset of part 1. However you should at least work out Part 1 in pseudo code before attempting part 2.

Your Tasks:

Part 1:
Fortunately, you (and your trusty laptop) happened to be in the neighbourhood and the three peace officers prevailed on you to write a program that they could use to “simulate” their problem and find a solution. By testing their solutions on your simulator they hope to avoid any nasty surprises.
Write a program that simulates this problem. A sample run of the program has been provided to give you direction. Each peace officer is represented by the letter 'P' and each convict by the letter 'C'.

Sample Run

```
The Convict River Crossing Simulation

Situation:
Boat on Bank: 1
People on Bank 1:  P  P  P  C  C  C
People on Bank 2:
Choices are:
1-move one Peace Officer                 2–move two Peace Officers
3-move one Convict                       4-move two Convicts
5-move one Peace Officer and one Convict   6-quit
What is your choice (enter 1 to 6): 5

Situation:
Boat on Bank: 2
People on Bank 1:  P  P  C  C
People on Bank 2:  P  C

Choices are:
1-move one Peace Officer                 2–move two Peace Officers
3-move one Convict                       4-move two Convicts
5-move one Peace Officer and one Convict   6-quit
What is your choice (enter 1 to 6): 3
```

```
Situation:
Boat on Bank: 1
People on Bank 1:  P  P  C  C  C
People on Bank 2:  P
DISASTER!!! The Convicts Overwhelm the Peace Officers!!
```

Question 3 - Part 1

Part 2:

Thanks to your program the peace officers and their charges cross the river without incident.

However the whole situation has intrigued you. As you watched the officers learn how to solve their problem by carrying out several trials with the simulation you wondered if a computer could learn in a similar fashion. Could the computer learn how to solve the puzzle by generating a choice, running the simulation to find out the consequences of that choice, keeping track of the results and then use this information to do better on subsequent trials?

You decide to try to modify your program to let the computer run a number of trials of the simulation in much the same manner as the police officers did. In each trial the computer generates a choice, processes the choice to see the consequences, records the results of those choices and uses that information on subsequent trials. Over the course of the trials the computer should come closer and closer to generating a set of successful moves.

Write an algorithm or program in which the computer "learns" how to solve the puzzle. Your algorithm/program should have the computer generate a choice from 1 to 5 (simulating choice by a human user), determine the consequences of that choice, keep track of the choice and the results and using that information in subsequent trials.

Use the following sample run for more cues on how to proceed.

Sample Run

Trial 1

```
The Convict River Crossing Simulation - AI Version

Situation:
Boat on Bank: 1
People on Bank 1:  P  P  P  C  C  C
People on Bank 2:

Computer's Choices are:
1-move one Peace Officer                   2–move two Peace Officers
3-move one Convict                         4-move two Convicts
5-move one Peace Officer and one Convict
The computer decides to do: 1

Situation:
Boat on Bank: 2
People on Bank 1:  P  P  C  C  C
People on Bank 2:  P

DISASTER!!! The Convicts Overwhelm the Peace Officers!!
```

In trial one of the sample run the computer "chose" option 1. That led to an unsuccessful result. The computer should record the choice and the results and avoid using option 1 on the first move on subsequent trials.

Trial 2

```
The Convict River Crossing Simulation - AI Version

Situation:
Boat on Bank: 1
People on Bank 1:  P  P  P  C  C  C
People on Bank 2:

Computer's Choices are:
1-move one Peace Officer                   2–move two Peace Officers
3-move one Convict                         4-move two Convicts
5-move one Peace Officer and one Convict
The computer decides to do: 2

Situation:
Boat on Bank: 1
People on Bank 1:  P  P  C  C  C
People on Bank 2:  P
DISASTER!!! The Convicts Overwhelm the Peace officers!!
```

In trial 2 the computer "reviews" past results and deliberately avoided using choice 1 as its first choice. Instead it opted for choice 2. However, this also led to disaster and choice 2 was added to the computer's list of moves do avoid for round one of the simulation.

Let's assume that the computer "chooses" choice 3 as its first move in trial 3. That move doesn't end the simulation but it is not overly successful because it only leaves the computer with one option on its next move and exercising that option basically brings the computer back to the beginning. So to be a successful learner the computer will need an additional mechanism to deal with moves that don't lose the simulation but don't result in any real progress towards the goal.

Note: There is no expectation that you will be able to write a fully functional program in the time you have available. We are interested in seeing how far you can get in the time allowed.

Question 3 - Part 2

Question 4: The Cyber Sleuth

You've recently been hired by your local city police department as a special consultant. Your task is to take clues provided to you by detectives and other officers and to write computer programs that will use these clues to solve the associated crime.

Important Note: If you decide to do Part 2 of this question you do not have to do Part 1. Part two is really a superset of part 1. However you should at least work out Part 1 in pseudo code before attempting part 2.

Part 1:
Your first assignment involves a tobacco store theft. The clerk had taken out a valuable meerschaum pipe from behind the counter to show a customer. The customer wasn't interested in the pipe and the clerk moved down the counter to show the customer some other merchandise. Much to his dismay when he returned to the cushion that had nestled the pipe he discovered it was gone.

Fortunately he had the presence of mind to leave the scene as it was and to call the police. The police determined the following:

1. Two other customers had been in the store at the time. From their descriptions they identified them as two veteran thieves - Johnny D and Sammy L.
2. Johnny D was famous for his long flowing red ponytail and Sammy L for his silver grey mullet. The two thieves hated one another so there was no chance that they had worked together on the theft.
3. The detectives found cigar ash on the cushion the pipe had been stolen from. As the clerk did not smoke, the ash had to have been left by the thief.
4. The clerk remembered that the red haired man had smoked a pipe

Your job is to write a program that uses these clues to identify the thief. You quickly determine that the data could be organized in the form of a table with the following structure…

Suspects and Clues

Suspect	**Johnny D**	**Sammy L**
Hair color		
Tobacco Smoked		

There is no input for this program. The clues are to be hard coded into the program using assignment statements and appropriate decision statements are to be used to fill in the cells of the table.

i.e. `if JohnnyDTobacco equals "Cigar" Thief equals "Johnny D"`

The output is to display a table such as the one above except with all of the cells filled in. Immediately below the table you are to have the statement identifying the thief. (Hint: First solve the puzzle using paper and pencil. Once you've done that analyze what you did, organize your actions into a step by step solution (an algorithm) and translate the algorithm into code.) **Remember your code must contain the logic required to solve the crime.**

Question 4 - Part 1

Part 2:

Your next assignment is a more demanding one. It involves a murder that took place only a few days ago on the 14th floor of a posh downtown hotel frequented by important people in the petroleum industry. The victim was a noted oil businessman, Mr. O. L. Khan.

The detectives have narrowed the field of suspects to four suspects, all who had adjoining rooms in the hotel. There is some confusion in the records but it has been established that the suspects occupied suites 1410, 1412, 1414 and 1416. The suspects are: Dr. Steh E. Scope, a local physician and investor, B. N. Counter, an oil company accountant, Colonel B.B. Gunn, a security specialist and Mr. Igneous Shale a petroleum geologist.
The clues are as follows:

1. Suites 1410 and 1416 are corner suites. .
2. Dr. Scope occupied Room 1410.
3. Forensic evidence indicated that the man occupying Room 1414 had black hair.
4. Either Colonel Gunn or Dr. Scope wore glasses.
5. Mr. Shale always wore a gold Rolex watch.
6. The man who wore glasses had brown hair.
7. Mr. Counter wore a large diamond ring.
8. The man in Room 1416 wore custom made shirts.
9. Mr. Counter occupied Room 1412.
10. The man with the custom made shirt had red hair.
11. The man in Room 1412 had grey hair.
12. The man with a gold Rolex occupied Room 1414.
13. Colonel Gunn occupied a corner room.
14. The murderer had brown hair.

Your job is to write a program that uses these clues to identify the murderer. You quickly determine that the data could be organized in the form of a table with the following structure.

Suspects and Clues

Suspect	**Dr. Scope**	**Mr. Counter**	**Col. Gunn**	**Mr. Shale**
Room Number				
Apparel				
Hair color				

There is no input for this program. The clues are to be hard coded into the program using assignment statements and appropriate decision statements are to be used to fill in the cells of the table.

i.e. `if SuspectHair(DrScope)equals "Brown" Murderer equals "Dr Scope"`

The output is to display a table such as the one above except with all of the cells filled in. Immediately below the table you are to have the statement identifying the murderer. (Hint: First solve the puzzle using paper and pencil. Once you've done that analyze what you did, organize your actions into a step by step solution (an algorithm) and translate the algorithm into code.) Remember your code must contain the logic required to solve the crime.

Question 4 - Part 2

Question 5 - Namorian Calculator

The Federation Starship Enterprise is in orbit around an inhabited planet called Namor. You happen to be a Science Officer on the ship. The Namoran have a pre-industrial culture.

The Namorans don't use a decimal number system. Instead, their system uses a small set of symbols to represent key quantities and then combines those symbols in a sequence to represent specific quantities. The Captain wants to demonstrate Federation technology to Namoran scholars and he has asked you to create a simple Namoran calculator.

He has asked you create a simple calculator which will accept two Namoran numerals and output the sum in Namoran numerals. As this is just a demonstration you are to assume that no numbers greater than decimal 1999 will be entered into the calculator and so no number greater than decimal 3998 will be generated. Here is an outline of the Namoran system…

1. The Namorans use the symbols **`I`**, **`V`**, **`X`**, **`L`**, **`C`**, **`D`** and **`M`**. Their decimal equivalents are as follows: **`I`** for 1, **`V`** for 5, **`X`** for 10, **`L`** for 50, **`C`** for 100, **`D`** for 500, and **`M`** for 1000.

2. Namoran numbers are formed by writing symbols from left to right. The symbols are read as a sum in which the value of each symbol is added together. By way of example, **`LV`** would translate as 55 in decimal notation.

3. In most cases Namoran numbers are written with the symbols representing the largest values on the left. For example in the Namoran number "**`LV`**" the **`L`** represents a larger value than the **`V`**. Notice this organization in the number **`CLI`**. The **`C`** represents decimal 100, the **`L`** represents decimal 50 and the **`I`** decimal 1.

4. However the symbols may only be used up to three times in succession. You cannot have a number such as **`CCCCL`** (an attempt to display 450 in Namoran). If the rules mentioned in points 1, 2 and 3 violate the three times in succession rule then the following rule must be used.

Numbers that would normally call for four of the same symbols in a row use a subtraction mechanism to shorten the length of the number. When a single **`I`** immediately precedes a **`V`** or an **`X`** it is subtracted. When a single **`X`** immediately precedes an **`L`** or a **`C`**, it is subtracted as well. The same occurs when a single **`C`** immediately precedes a **`D`** or an **`M`**. The value of **`C`** is subtracted from the value of the **`D`** or the **`M`**.
So to represent decimal 4 one would use **`IV`** rather than **`IIII`**, to represent 9 one would use **`IX`** rather than **`VIIII`**. This can be complex. Numbers such as 49 in decimal would be represented by **`XLIX`** in Namoran.

Your Tasks:

Important Note: If you decide to do Part 2 of this question you do not have to do Part 1. Part two is really a superset of part 1. However you should at least work out Part 1 in pseudo code before attempting part 2.

Part 1:
The captain just wants a simple prototype to demonstrate the conversion of decimal numbers into Namorian numbers. Your program should allow for the user to enter a set of decimal numbers which it will translate into its Namorian equivalent. For this prototype no number greater than 100 will be used. The user will enter a **`Q`** (for quit) when all numbers have been entered.

Sample Input/Output

```
  Enter your numbers:
  4         4 = IV
  60        60 = LX
  90        90 = XC
  Q         DONE
```

Question 5 - Part 1

Part 2:

Now that you've demonstrated that the prototype works the captain wants a more sophisticated calculator - working completely in Namorian numbers. Your program should allow the user to:

1. enter two Namoran numbers followed by an equal sign,
2. calculate the sum,
3. print out the sum,
4. repeat the process until **Q** (for Quit) is entered.

Input will be from the keyboard and output to the screen. Follow the format in the following sample display. In the sample display the input data in is shown as underlined.

Sample Input/Output

```
  Enter your numbers:
  II+IV= VI
  LX+XCIII= CLIII
  D+CDXCVIII= CMXCVIII
  Q
```

Done

Note: The decimal equivalents of the above expressions are:
II+IV= VI is equivalent to **2+4= 6**
LX+XCIII=CLIII is equivalent to **60+93= 153**
D+CDXCVIII= is equivalent to **500+498= 998**

Question 5 - Part 2