UNIT TEST REVIEW:
FOUNDATIONS OF PROGRAMMING I

This unit test will include everything we covered in **UNIT 3: Foundations of Programming I**, including decision structures, handling exceptions, randomizing numbers, the **ListBox** control and loop structures.

PART I: KNOWLEDGE AND UNDERSTANDING *(20 MARKS)*

- Multiple Choice

PART II: THINKING *(20 MARKS)*

- Review how to use a **Random** object to randomly generate a random number.
- Review how to write a **For…Next** loop that repeatedly outputs information to a **ListBox** control.
- Review how a program that uses decision structures works.
- Review how to write a **try-catch** statement that handles an exception that is thrown when the user enters invalid data into a text box.
- Review how to write a confirm dialog box that asks the user YES/NO question.

PART III: COMMUNICATION *(20 MARKS)*

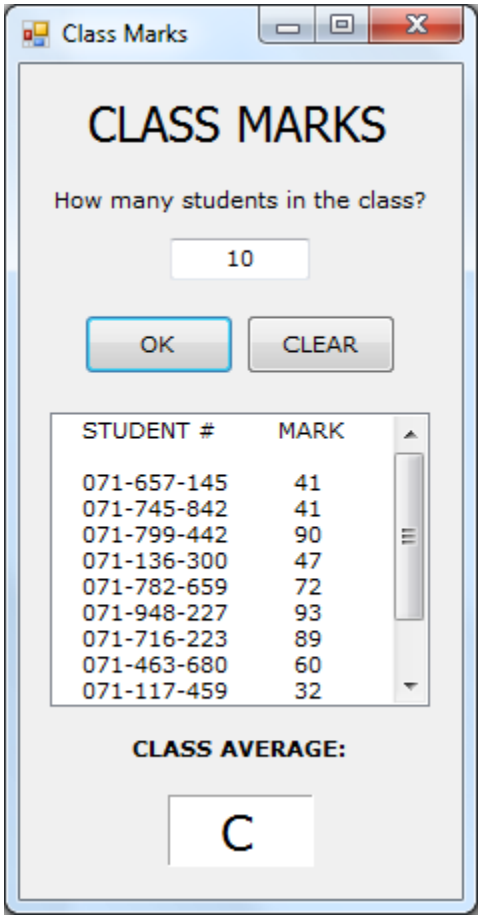
- Review the function(s) of each of the flowchart symbols that we covered in class.
- Review the difference between **global** and **local** variables, including where they are declared, how they are declared, and the scope of the variable. You'll need to go back to the handout on **Variables** in UNIT 2, Lesson 5.
- Review the logical operators in terms of how they are used in Boolean expressions (i.e. **And, Or, Xor, Not**).
- Review the difference between a **counted** loop and a **conditional** loop.
- Review the difference between a **Do While…Loop** and a **Do…Loop While**.
- Review the difference between the **Not** logical operator and the **< >** relational operator.

PART IV: APPLICATION *(20 MARKS)*

- Review how to write a program that uses decision structures, loops, the **Random** object and the **ListBox** control.

PROGRAMMING EXERCISE

1. Create a **Class Marks** program that prompts the user for the number of students in the class. When the user clicks **OK**, the program should randomly generate 9-digit student numbers that begin with 071 and marks between 30 and 100 inclusive, and output the student numbers, marks and class average (rounded to 1 decimal place) in a **ListBox** control.



Your program must also implement a decision structure to convert the class average into a letter grade according to the following conditions:

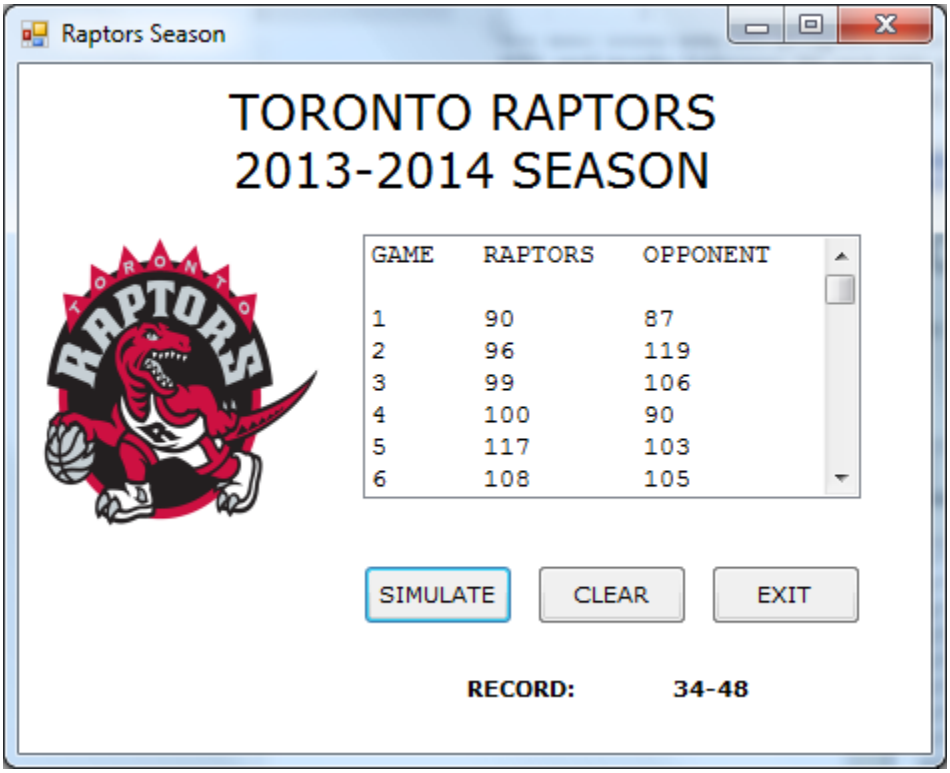
NUMERIC GRADE	LETTER GRADE
>= 80	A
70 – 79	B
60 – 69	C
50 – 59	D
< 50	F

Once the class average has been converted to a letter grade, it should be outputted in the label called **lblAverage**.

Be sure to also use a **try-catch** statement to handle the exception that may get thrown if the user enters bad data in the text box (e.g. a number with decimals, a String value, a null value, etc.).

Save the program in a folder called **Class Marks** in your UNIT 3 folder.

2. Create a **Raptors Season** program that simulates an 82-game schedule by randomly generating and outputting two scores between 85 and 120 when the user clicks the **SIMULATE** button. One score represents the total points the Raptors scored in the game while the second score will represent the points their opponents scored in the game. The scores should be outputted in a **ListBox** component as follows:



As the program generates each game result, it must calculate the Raptors' record by adding the number of wins and the number of losses. Once the simulation is complete, the Raptors' record should be outputted in a label and one of the following messages should be outputted to the user in the form of a message box:

If the Raptors end up with a winning record, the message should be "WOO HOO…the Raptors had a winning season!"

If the Raptors end up with a losing record, the message should be "BOO…the Raptors suck!"

When the user clicks the **CLEAR** button the list box and the record should be cleared.

BONUS:

Modify your program by implementing a **While** loop so that if any of the games end in a tie, the program will re-simulate the game until it's not a tie.

Save the program in a folder called **Raptors Season** in your UNIT 3 folder.