**Institute of Technology Tralee**

**Ord/Hons BSc. in Computing with Specialism (Group C) - Year 1**

**Continuous Assessment #2**

**Date: 2/12/11**

**Time: 9 – 11 a.m.**

**Introduction to Programming**

**Instructions:** Attempt the following question. You should use the Just BASIC IDE for coding. When you are finished you must print out your code for correction.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Q1.**

Each Irish citizen is allocated a PPS number once they reach the age of 18 that is unique to them and used for taxation purposes. A valid PPS number will begin with exactly 7 digits and end with one or two uppercase letters. So, for example, 1234567A would be considered a valid PPS number, as would 7863456RT, but 6478TY\*%& and 8768086b would both be considered invalid PPS numbers.

You must write a Just BASIC program that processes an arbitrary number of PPS numbers. End of input is signaled by the user entering “q” and you should use a **do-until** loop for the main loop iteration process here. As each PPS number is entered it must be fully validated so that it complies with the specifics outlined in the previous paragraph. As long as the PPS number entered remains invalid, it should be rejected and the user prompted to re-enter. You can use **while-wend** loops here for validation purposes and any other structures you wish.

The program should also keep track of the total number of valid PPS numbers that have been entered.

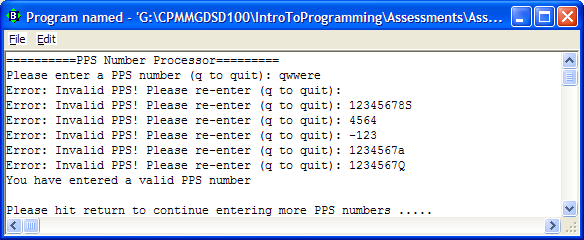
In coding your solution, you should find the **asc**(), **mid$**() and **len**() functions to be particularly useful.

Using the test values as indicated in the screen shots below, the program should give you **exactly** the following output when it runs, including banners, blank lines, cleared screens etc.

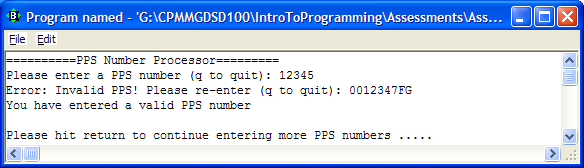
Also note that there will be a few marks awarded for having a **meaningful comment at the top of the program** and for ensuring that your program **terminates properly** by handing back all the resources it has used to the system on completion.

**Sample Screen Shots**

**In this first iteration of the main loop, the user is prompted for a PPS number. As long as the value entered does not comply with the definition of a valid PPS number (or if the user just hits return), the value will be rejected and the user asked continuously to re-enter until a valid PPS number is supplied. When a valid PPS number gets supplied the user receives a confirmation message and then the program halts until the user hits return on the keyboard**



**Once the user hits return, the screen clears and the second iteration of the main loop begins. The user is prompted for another PPS number and the validation process continues as before. Again, as soon as a valid PPS number is entered the confirmation message appears and the program halts until the user hits return.**



**When the user has entered all the PPS numbers they wanted to, they enter “q” to quit the main loop and then the program displays the total number of valid PPS number that were entered.**

