**Declaration on Plagiarism**

**Assignment Submission Form**

**This form must be filled in and completed by the student(s) submitting an assignment**

**Name(s):** John Griffin

**Programme:** CA

**Module Code:** CA208

**Assignment Title:** Assignment Two

**Submission Date:** 11/12/2017

**Module Coordinator:** David Sinclair

**I/We declare that this material, which I/We now submit for assessment, is entirely my/our**

**own work and has not been taken from the work of others, save and to the extent that such**

**work has been cited and acknowledged within the text of my/our work. I/We understand**

**that plagiarism, collusion, and copying are grave and serious offences in the university and**

**accept the penalties that would be imposed should I engage in plagiarism, collusion or**

**copying. I/We have read and understood the Assignment Regulations. I/We have identified**

**and included the source of all facts, ideas, opinions, and viewpoints of others in the**

**assignment references. Direct quotations from books, journal articles, internet sources,**

**module text, or any other source whatsoever are acknowledged and the source cited are**

**identified in the assignment references. This assignment, or any part of it, has not been**

**previously submitted by me/us or any other person for assessment on this or any other**

**course of study.**

**I/We have read and understood the referencing guidelines found at**

**http://www.dcu.ie/info/regulations/plagiarism.shtml , https://www4.dcu.ie/students/az/plagiarism**

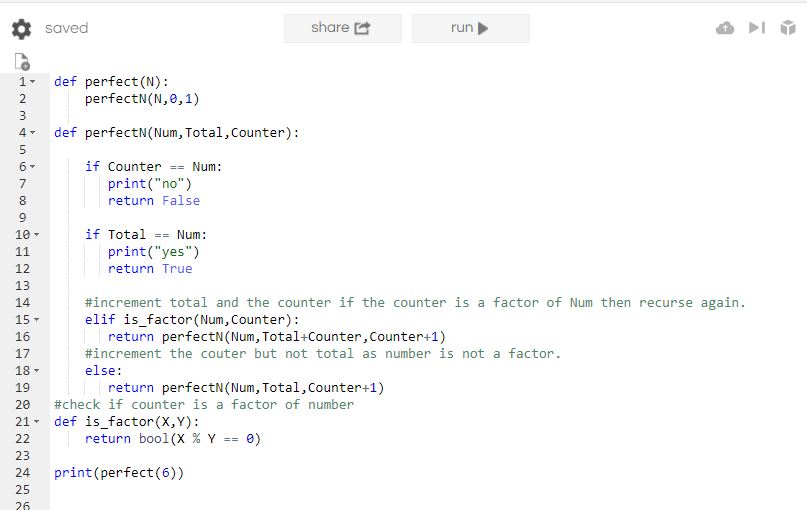
**and/or recommended in the assignment guidelines.**

**Name(s): \_\_\_\_\_John Griffin\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_11/12/2017\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Understanding the Problem posed.**

**Question - Write a Prolog predicate perfect(N) that returns true if N is a perfect number. A number, N, is a perfect number if the sum of its factors, not including itself, is equal to N. For example, 6 is a perfect number. Its factors, not including itself, are 1, 2 and 3; and their sum is 6.**

My initial reaction to the problem was that, as in the question, that a number is Perfect if the sum of its factors were equal to the number itself. The first thing I did when I understood was to try and get a better grasp on the underlying concepts behind it by writing the problem in a language I was a little bit more familiar with – Python. My initial approach was to use basic loops etc. but I soon realised that Prolog would only support a recursive approach, so I changed my solution to a recursive one. I was also aware that Prolog outputs two values – True and False so I made my python program as translatable into Prolog which would make the process much easier for me. My rough Python solution was as followed:



I felt that this was reasonably efficient and was most importantly correct. It was also very translatable into Prolog and was a good place to start to try and come up with a solution in Prolog.

The program was correct for the cases I could find so I was confident that this was a solution I could roughly base my Prolog solution on.

**Assumptions Made:**

The assumptions I made while writing my program where as followed:

* That the answer would return as True if Total was equal to the Number.
* If Count divides into the Num equally, it is a factor.

**My Solution**

The first part of the function I wrote was the predicate that would be called from the Prolog terminal, and its argument contains the Integer N which the user inputs. I then decided to find a way to call up another predicate so I could begin the recursion aspect of the question. I used another predicate(pNumber) because I could also add 0 as the Total Value and 1 as my counter, similar as to my python solution. This would allow me to track the Total of the Factors easily while recursing through 1…N. I then had to write the base case of the question. This will return to True to perfect when the Total is equal to N after count has reached the number N, or else it Fails and returns False.

After this I needed to find a way to check if the Counter was a factor of Num and if so, I would add Counter to Total and then increment it. The way I found the clearest and easiest to do was to add another predicate isafactor which would take in the counter and number as its arguments, make sure that the counter is less than the number and return True if they satisfied the condition. This was by far the most challenging part of the exercise for me because at first I did not include the fact that count must be less than the Number. This was especially important for numbers such as 1 as it caused infinite loops. The Count could then be added to the Total and then incremented once this returned True. As in the python question, I then called up the base predicate again (perfectN) with the same number (to compare against) but with the new values for Count and Total.

If the predicate above returned False as Count is not a factor of N, I designed the program so that Count would still increase in value by one. To ensure this happened, I put a predicate called notfactor inside this to help make the algorithm clearer and also to ensure, once again, that Count is not greater than Num. If the number was not a factor, then count would be incremented while the Total value remained the same as Count was not a factor match.