COMP 1405Z – Fall 2023
Introduction to Computer Science I
Midterm – Wednesday, October 4th 2023

Midterm Details

The midterm will be submitted through Brightspace. Put your solutions to each of the problems specified in this document in the corresponding .py file from the provided 1405-midterm-materials.zip file. To submit your midterm solutions, create a single .zip file with all of your midterm materials and submit it to the '1405 Midterm' assignment submission on Brightspace. The official end time for the midterm is 2:25pm (Eastern Time) but you will have until 2:35pm (Eastern Time) to submit your midterm. **Do not wait until the last minute – no late submissions will be accepted**. After submitting, you should download your own zip file, extract its contents, and ensure the files have the correct content. You are not allowed to consult other students, TAs, or anybody else while completing the midterm. You may use any IDE or text editor you generally use to write your code. You can execute your programs as many times as you want to debug and fix your code.

Problem 1 (10 marks)

Write the code for this part in the problem1.py file. Write a function called largest_prime_factor that takes a single integer input argument N. You can assume that N > 0. The function must return the largest positive integer X such that:

1. X evenly divides into N. That is, N/X produces no remainder.
2. X is prime (i.e., only positive integers that evenly divide into X are 1 and X).

For example, if N is 9, the function would return 3, since 3 is prime, divides into 9 evenly, and no larger integers meet the requirements. If N was 31, the function would return 31, since 31 is prime and divides into itself once with no remainder. A problem1-tester.py file is provided to facilitate testing your function.

Problem 2 (15 marks)

Write the code for this part in the problem2.py file. Write a function called *analyze(filename)*, which accepts a single string argument representing the name of a file. You can assume the file name you are given will represent a file that contains a single positive integer value on each line. The function must return the length of the longest consecutive sequence of odd numbers inside the given file. For example, if the numbers in the file were 2 4 1 6 8 3 5 8 2 **1 3 7** 6 5, the function should return 3 because the bolded sequence is 3 odd numbers in a row and no longer sequence of odd numbers exists. A problem2-tester.py file is provided to facilitate testing your function.

Problem 3 (25 marks)

Write the code for this part in the problem3.py file. **You may not use additional modules, lists, dictionaries, or any other form of expandable storage other than file input/output for this problem.** In other words, you should solve this problem using only the course concepts that have been covered up to and including functions (i.e., variables, branches, loops, files, functions). **You CAN use the string.strip() method**. Write a function called print_sorted_grades(filename) that accepts a single string argument representing the name of a file. You can assume that the file with the given name will contain data in the following pattern:

```
Student1_first_name
Student1_last_name
Student1_student_number
Student1_assignment_grade
Student1_midterm_grade
Student1_exam_grade
Student2_first_name
Student2_last_name
Student2_student_number
Student2_assignment_grade
Student2_midterm_grade
Student2_exam_grade
...etc...
```

You can assume that the student numbers are integers and all student numbers within the file are unique. Using this information, the final grade for a student can be calculated with the following weights: assignment=25%, midterm=30%, exam=45%. Your function must print to the console the name and final grade for each student contained in the file in descending order of final grade (i.e., highest grade student to lowest grade student). The information should include the student's full name and their grade. Note that it may be beneficial to create additional functions and/or files to complete this problem successfully. Remember to apply the computational thinking principles. If you cannot complete the entire problem, you may receive partial marks for successfully completing parts of the overall solution.

Example data is provided in the five studentinfoX.txt files, so you can test your implementation. The problem3-expected-output.txt file contains the expected output for each of the provided studentinfoX.txt files. The grades in the expected output have been rounded to 2 decimal places. You are not required to round the grades in your solution.