TIC1001—Introduction to Coputing and Programming
National University of Singapore
School of Continuing and Lifelong Education

Practical Examination 1

23 September 2017

Time allowed: 2 hours

Instructions (please read carefully):

1. This is **an open-book exam**. You are allowed to bring in any course or reference materials in printed form. No electronic media or storage devices are allowed.
2. This practical exam consists of **two** questions. The time allowed for solving this test is **2 hours**.
3. The maximum score of this test is **20 marks**. Note that the number of marks awarded for each question **IS NOT** correlated with the difficulty of the question.
4. You are advised to attempt all questions. Even if you cannot solve a question correctly, you are likely to get some partial credit for a credible attempt.
5. While you are also provided with the templates `q1-template.c` and `q2-template.c` to work with, your answers should be submitted on Coursemology. Note that you can **only run the test cases on Coursemology for a limited number of tries** because they are only for checking that your code is submitted correctly. You are expected to test your own code for correctness locally on VS Code and not depend only on the provided test cases. Do ensure that you pasted your answers correctly by running the test cases at least once.
6. In case there are problems with Coursemology.org and we are not able to upload the answers to Coursemology.org, you will be required to name your file `q1-<student no>.c` and `q2-<student no>.c` where `<student no>` is your student number and leave the file in the `tic1001-pe1` folder on the Desktop. If your file is not named correctly, we will choose any .c file found at random.
7. Please note that it shall be your responsibility to ensure that your solution is submitted correctly to Coursemology and correctly left in the desktop folder at the end of the examination. Failure to do so will render you liable to receiving a grade of **ZERO** for the Practical Exam, or the parts that are not uploaded correctly.
8. Please note that while sample executions are given, it is **not sufficient to write programs that simply satisfy the given examples**. Your programs will be tested on other inputs and they should exhibit the required behaviours as specified by the problems to get full credit. There is no need for you to submit test cases.

ALL THE BEST!

Question 1: Numbers in love! [10 marks]

Mathematician Dr. Wink discovered pair of numbers that are "deeply in love" with each other and decided to name them "lovely numbers".

He asked for your help to computerize the procedure to identify these "lovely numbers".

A. Sum of all factors

As a start, write a function `int sum_of_factors(int n)`. This function returns the sum of all integer factors (including 1 but excluding n itself) for the number n. Note: $factor$ of a integer n refers to integers that can divide n completely without remainder.

For example, given $n = 10$, this function returns $1+2+5 = \mathbf{8}$. You can assume $n \geq 1$ and the sum of factors will not exceed the maximum `int` bound.

[5 marks]

B. Lovely numbers

Dr.Wink tells you that if two numbers $n1$ and $n2$ meet the following criteria:

- The sum of all factors for $n1$ equals to $n2$
- The sum of all factors for $n2$ equals to $n1$

then $n1$ and $n2$ are said to be a pair of **lovely number**.

For example, given $n1 = 284$ and $n2 = 220$:

- The sum of all factors for 284 is $1+2+4+71+142 = \mathbf{220}$
- The sum of all factors for 220 is $1+2+4+5+10+11+20+22+44+55+110 = \mathbf{284}$

Hence, 284 and 220 is a pair of lovely numbers.

Write a function `bool is_lovely(int n1, int n2)` to perform this check. You should reuse the function `sum_of_factors`.

[5 marks]

Question 2: Loan [10 marks]

Uncle Soo needs to take out a loan to get a large supply of CPUs to build a supercomputer. To avoid compounding interest, he decides to make a fixed repayment every month to pay off the loan. This is known as an amortized loan.

One can easily compute the loan using derived formulas, but the idea to compute it is simple: every month, an interest is charged upon the outstanding loan amount (known as the principal) and Uncle Soo makes a payment to pay off the interest and some of the principal. This continues until the entire loan is paid off.

For example, suppose Uncle Soo takes a loan amount of $1,000,000 with a monthly interest rate of 0.5% and he wishes to make a monthly payment of $6,000.

At month 1, the interest of the loan would be $\$1,000,000 \times 0.5\% = \$5,000$. Uncle Soo pays $6,000 so the remainder of the principal is $\$1,000,000 + \$5,000 - \$6,000 = \$999,000$.

In month 2, the interest would be $0.5\% \times \$999,000 = \$4,995$, and the principal is reduced to $\$999,000 + \$4,995 - \$6,000 = \$997,995$.

We can summarize it a simple table:

Month	Principal	Interest	Payment	Remaining
1	$1,000,000.000	$5,000.000	$6,000	$999,000.000
2	$999,000.000	$4,995.000	$6,000	$997,995.000
3	$997,995.000	$4,989.975	$6,000	$996,984.975
4	$996,984.975	$4,984.925	$6,000	$995,969.900
...	...	...	...	...
359	$7,440.368	$37.202	$6,000	$1,477.570
360	$1,477.570	$7.388	$1,485	$0.000

As you can tell, it will a long time, 30 years to be exact for Uncle Soo to pay off the loan.

Variable Interest Rate

Uncle Soo is lucky if he finds a loan with a fixed interest rate. It is possible for the interest rate to change as the loan is serviced. One simple scheme is to increase the interest by a constant factor after every 12 months.

For example, if the factor is 1.05, then using the same example, the interest will be increased to $0.5\% \times 1.05 = 0.525\%$ for the 13th month, and to $0.525\% \times 1.05 = 0.5513\%$ for the 25th month and so on.

Unfortunately for Uncle Soo, under this scheme he will not be able to ever finish off paying the loan with $6,000 per month as the interest would be increased to 0.6078% after 5 years and the interest incurred at that point is higher than the monthly payment!

Task: Help Uncle Soo determine how many months it will take for him to finish off paying a loan.

Implement the function `int num_payments` that takes the following inputs:

- `double principal` which is the initial amount of the loan.
- `double interest` which is the monthly interest rate.
- `double factor` which is the factor that the interest rate will increase every 12 months.
- `double payment` which is the monthly payment.

Assuming that payment is made consistently every month, the function should output the number of months required for the loan to be paid off. In the event that the loan can never be paid off, i.e., the monthly interest is higher than the payment, `-1` is returned.

You may also assume that the maximum `int` bound will not be exceeded in the computations.

Note: You are not allowed to use a formula to simply compute the loan repayment period. You will have to solve it computationally.

Advice: You might wish to ensure you correctly implement the basic functionality first, such as ignoring the factor (i.e. factor = 1), before attempting to handle additional requirements.

[10 marks]

Sample tests:

Test function	Output
`num_payments(1000000, 0.005, 1.0, 6000)`	`360`
`num_payments(1000000, 0.005, 1.01, 6000)`	`449`
`num_payments(1000000, 0.005, 1.05, 6000)`	`-1`

— E N D   O F   P A P E R —