Assignment 1.1

Computational thinking, algorithms, flowcharts, and pseudocode

---

SUBMISSION REQUIREMENTS: Submit a single PDF file called **assignment1.1.pdf** to cuLearn. You should create your solutions for this assignment using Microsoft Word, Google Docs, or something similar, and save the document as a PDF when you are finished. If you are having trouble creating a PDF, visit office hours or post on cuLearn. Instructions for getting Microsoft Office (including Word) for free at: https://carleton.ca/its/ms-offer-students/

This assignment has **30 marks**.

LATE SUBMISSIONS: Submissions can be submitted up to 8 hours after the deadline. For each hour it is late, the assignment will receive a 2.5% penalty. No submissions will be accepted after this period for any reason. You are allowed as many resubmissions as you would like, but we can only see (and will only grade) the final submission.

INVALID SUBMISSIONS: Submissions with an incorrect name or that are not a PDF will receive a **-5 mark penalty**, with no exceptions. You are responsible for following the guidelines outlined in this assignment and in the course outline. If the TA **can not understand your writing** or solution, you may receive no marks for that section. Write clearly.

GRADE DISPUTES: If you feel you were graded incorrectly, you have seven (7) days to contact the TA that graded your assignment to have it corrected. After this, corrections will not be considered.

NOTE: This is part one of a two part assignment totalling 50 marks. Your final grade will include these marks (0-30) plus the marks of A1.2 (0-20). Invalid submission penalties will only be applied once. Late submission penalties and deadline cutoffs **still apply to both assignments** (eg. if you submit A1.1 two hours late, A1.1 will receive a 5% reduction (5% of 30 marks is 1.5 marks); if you submit A1.2 *four* hours late, A1.2 will receive a 10% reduction (10% of 20 marks is 2 marks) - thus you will receive a total of 3.5 marks off of the assignment.

Problem 1 of 4 (Computational Thinking, 6 marks)

Marking Scheme: 0.5/cornerstone, up to 1/example

In the lecture, we discussed the four cornerstones of computational thinking (for example, decomposition).

For each of these cornerstones, provide one or two sentences about how you could apply that technique to help solve a problem. The problem should be from something you're interested in, and doesn't necessarily need to be about programming.

You may use the same example problem for all of the cornerstones, such as walking through the problem with each step, but you can not use examples that were already discussed in the lectures.

Problem 2 of 4 (Guessing Game Flowchart, 10 marks)

Marking Scheme: Up to 5 for correctly solving, Up to 2 for correct symbols, Up to 3 for a clear solution

It's time to plan! In this problem, you will be creating the flowchart for a basic higher/lower guessing game. The "program" will start by picking a random number (assume this is a simple operation) and then ask the user to make a guess. If the user guesses incorrectly, the program should output a hint which tells them to guess higher or to guess lower.

Once the user guesses correctly, the program should output "You win!" and end. Use the flowchart symbols that were described in the lectures. You can choose to use a computer program to create flowcharts or draw neatly on paper.

Your goal for this problem is to create the flowchart for a guessing game, in which a player must repeatedly guess numbers until they have guessed a secret number (assume the secret number is 42). If the user guesses incorrectly, the user should get a hint indicating "guess higher" or "guess lower". When the user has guessed the number correctly, the 'program' should output "You Win!" before ending. Your answer should use the flowchart symbols outlined in the lectures. You can use a computer program to create the flowchart, or write it neatly on paper, take a picture, and include it in your document.

Problem 3 of 4 (Guessing Game Pseudocode, 10 marks)

Marking Scheme: Up to 5 for correctly solving, Up to 2 for correct format (tabs, numbering), Up to 3 for a clear solution

Instead of using a flowchart, create a solution to the previous problem using pseudocode. Your pseudocode should resemble that used in lectures, but as long as your meaning is clear/unambiguous, you are free to use any words you desire.

Problem 4 of 4 (Testing the Plan, 4 marks)

Marking Scheme: Up to 2 for all steps shown correctly, Up to 2 for clear formatting

Now is your chance to test that your algorithm works. Using either the flowchart or the pseudocode, walk through your plan with with a scripted, sample input.

Assume the random number generated is 42. The user will guess: 10, 20, 50, 25, then 42 to end the program.

You may show the steps of your walkthrough any way you like. I recommend labelling the steps of your plan (flowchart or pseudocode) with numbers to refer to them easily, like shown in the lectures (1.1, 1.2, 2, 3, 1). Try to show how the numbers change, and the result of comparisons (**if**).

Note: This section doesn't need to be perfectly formatted. Just show that you've worked through the plan with an example. You can also **clearly** write this on paper, if it is easier, and take a picture to submit.