Exam Information

The online exam will be held from 9:00 a.m. to 12:00 noon on Saturday, April 11.

You will access it through the Moodle page for the course. For some questions, the answers will be typed directly into Moodle. For some questions, you will also be given the option of uploading a file (for example, you might want to include a photograph of a diagram). For uploaded files, formats permitted will be:

- PDF (preferred)
- GIF JPEG PNG TIFF image files
- Word processor files (Open Document Text .odt; Word document .docx or .doc)

Pre-Exam Exercises

Two of the questions on the exam will be related to the material in this document.

Important: Prior to the exam, you may discuss the contents of this document in general terms with others. However, you should not discuss them in detail. You must not take any written notes away from such discussions, and you must not write anything down for at least one hour after such a discussion. (This is for your own protection, to ensure that any solutions you write on the exam will not look similar to other students' solutions. If your solution to an exam question looks unusually similar to another student's solution, you could be charged with academic dishonesty.)

During the exam on Saturday and while writing your code for question 2, below, you must not communicate in any way with other humans, except the instructor.

I expect students to spend about an hour on each of the two questions given here prior to Saturday. The rest of the time that you can devote to EECS3101Z should be spent studying for the exam as usual.

1. Consider the following algorithm.

```
1  FindDuplicate(A[1..n])
2      Precondition: n ≥ 2 and A is an array of n integers sorted in non-decreasing order with
3                    A[1] = 1 and A[n] = n − 1
4      lo ← 1
5      hi ← n
6      loop
7          exit when hi = lo + 1
8          m ← ⌈(lo+hi)/2⌉
9          if A[hi] − A[m] < A[m] − A[lo] then
10             lo ← m
11         else
12             hi ← m
13         end if
14     end loop
15     Postcondition: A[lo] = A[hi]
16 end FindDuplicate
```

Spend some time to figure out *in detail* how and why it works. One question on the exam will be related to this algorithm.

Hint 1: You may want to think about what the minimum and maximum possible number of *distinct* values there can be in an array A that satisfies the pre-conditions.

Hint 2: if $A[hi] - A[m] < A[m] - A[lo]$, notice that $2(A[hi] - A[m]) < (A[hi] - A[m]) + (A[m] - A[lo])$.

2. Mrs. Vilve Yachke is expanding her chain of restaurants to Toronto, since she knows there is a high demand for good quality cabbage rolls and coffee in this city. She plans to put all of her restaurants along Yonge Street, since she wants her customers to have convenient access using the subway.

She divides Yonge Street into $n \geq 2$ segments numbered $0..n-1$ from the waterfront to Steeles. She will build at most one restaurant in each segment, and she will not build two restaurants in two adjacent segments (to avoid oversaturating the market). Her real estate agent tells her that the cheapest property in segment i costs c_i dollars for $i = 0..n-1$. Her total budget for buying properties is B dollars. All prices and B are positive integers.

Design a dynamic programming algorithm to compute the maximum number of properties she can buy for her restaurants.

A stub of Java code is given on the course Moodle page in the file `Yachke.java`. When this code is run, a user can type an input on a line in the following format:
B n c_0 c_1 ... c_{n-1}
and the programme will output the maximum possible number of restaurants. The programme is designed to process multiple inputs of this form, one per line. It terminates when a value of 0 is entered for B.

Fill in code for the function `computeMax` using your dynamic programming solution. It is possible to do this using about a dozen lines of code. You shouldn't change anything outside of the `computeMax` routine.

You will also find sample input and output files `Yachke.in` and `Yachke.out` on the course Moodle page. You may test your solution by running

```
javac Yachke.java
java Yachke < Yachke.in > myoutput
diff myoutput Yachke.out
```

If your implementation is correct, the `diff` command will produce no output. (The converse of this statement is false.) Once you are satisfied with your Java programme, submit your modified `Yachke.java` file using the same method you used to submit assignments 8, 9 and 10. You must do this before noon on Saturday, April 11.

On the exam, you will be asked about the algorithm you used.