University of Waterloo
CS240 - Spring 2020
Assignment 1

Due Date: Wednesday June 3, 5pm

The integrity of the grade you receive in this course is very important to you and the University of Waterloo. As part of every assessment in this course you must read and sign an Academic Integrity Declaration and submit to MarkUs within 48 hours of the release of the assessment to students and preferably before you start working on the assessment – i.e. **read, sign and submit A01-AcInDe.txt now or as soon as possible**. The agreement will indicate what you must do to ensure the integrity of your grade. If you are having difficulties with the assignment, course staff are there to help (provided it isn't "last minute").

The Academic Integrity Declaration must be signed and submitted on time or the assessment will not be marked.

Please refer to the course webpage for guidelines on submission. Submit your written solutions electronically as a PDF with file name a1Submit.pdf using MarkUs. We will also accept individual question files named a1q1.pdf, a1q2.pdf, ... , a1q6.pdf if you wish to submit questions as you complete them.
In this assignment, log is the base-2 logarithm.

Problem 1 [3+3+3+4+4 marks]

Provide a complete proof of the following statements from first principles (i.e., using the original definitions of order notation).

1. $5n^3 + n^2 + 10n + 7 \in O(n^3)$
2. $n^2(\log n)^{1.00001} \in \Omega(n^2)$
3. $\frac{n^2}{n+\log n} \in \Theta(n)$
4. $n \log(\log(n)) \in o(n(\log(\log(n)))^2)$
5. $2^n \in \omega(n^{1/n})$

Problem 2 [3+3+3 marks]

For each pair of the following functions, fill in the correct asymptotic notation among Θ, o, and ω in the statement $f(n) \in \sqcup(g(n))$. Provide a brief justification of your answers. In your justification you may use any relationship or technique that is described in class.

1. $f(n) = n\sqrt{n}$, $g(n) = n \log n$

2. $f(n) = 10^n + 99n^{10}$, $g(n) = 75^n + 25n^{27}$

3. $f(n) = n^4$, $g(n) = n^4(\log(n))^3$

Problem 3 [3+3+3+3+4 marks]

Analyze the following pieces of pseudocode and give a tight (Θ) bound on the running time as a function of n. Show your work. A formal proof is not required, but you should justify your answer (in all cases, n is assumed to be a positive integer).

1.
```
x = 0
for i = 1 to n + 12 do
   x = x * 4
   for j = 389 to 20100
      for k = 2i to 3i
         x = x * 77
```

2.
```
 x = 0
 for i = 1 to ceiling(log(n))
    for j = 1 to i
       for k = 1 to 10
          x = x + 1
```

3.
```
s = 0;
for i = 1 to n*n do
   for j = 1 to i*i do
        s = s + 1;
```

4.
```
s = n
while (s > 0)
  if (s is even)
    s = floor(s / 4)
  else
    s = 2*s
```

5.
```
i = 2
x = 0
while (i < n)
  for j = 1 to n
    x = x + 1
  i = i * i * i
```

Problem 4 [2+6 marks]

We consider two algorithms, Algo1 and Algo2, that solve the same problem. For any input of size n, Algo1 takes time $T_1(n)$ and Algo2 takes time $T_2(n)$. Prove or disprove each of the following statements. To prove a statement, you should provide a formal proof that is based on the definitions of the order notations. To disprove a statement, provide a counter example and explain it.

1. Suppose that $T_1(n) \in O(n^2(\log n)^5)$ and $T_2(n) \in O(n^3(\log n)^2)$. Does it imply that there exists n_0 such that for $n \geq n_0$, Algo1 runs faster than Algo2 on inputs of size n?

2. Same question, assuming that $T_1(n) \in \Theta(n^2)$ and $T_2(n) \in \Theta(n^3)$.

Problem 5 [2+4+4+6+6 marks]

In this problem, we consider the problem of merging several sorted arrays.

Suppose that we have a family $F = \{a_1, \ldots, a_s\}$ of sorted arrays to merge. Our strategy is to choose two of them, say a_i and a_j, remove them from F, merge them and put the resulting array back into F; we keep doing that until there is only one array in F. We will use a greedy strategy: at each step, we choose a_i of minimal length in F, and a_j of minimal length in $F - \{a_i\}$.

1. What data structure would you use to store F?

In all the following questions, we only count the number of comparisons we do to merge arrays; we do not count the cost of finding and removing a_i and a_j from F, and of inserting the merged array back into F. We are only interested in giving worst-case estimates.

2. Explain why we can merge two sorted arrays of lengths n_1, n_2 using in the worst case $n - 1$ comparisons, with $n = n_1 + n_2$ (hint: see the slides).

3. Suppose that we have three arrays a_1, a_2, a_3 to merge, of respective lengths $1, 2, 4$. Verify that this greedy strategy minimizes the worst case number of comparisons.

4. Suppose that we have s arrays of the same length k. Explain how the algorithm will run, and give a Θ estimate on the worst case number of comparisons it does, in terms of k and s. For simplicity, you can assume that s is a power of two.

5. Repeat the previous question, with the alteration that we have s arrays of lengths $1, 2, 4, 8, \ldots, 2^{s-1}$ (give the bound in terms of s).

Problem 6 [5+5 marks]

True or false? For each of the following assertions, indicate whether it is true or false. If true, prove it; if false, give a counter-example and briefly justify it. In both cases, f and g are functions that take positive values, and with $\lim_{n\to\infty} f(n) = \lim_{n\to\infty} g(n) = \infty$.

1. If $f(n)$ is $\Theta(g(n))$ and $h(n) \in \Theta(g(n))$, then $\frac{f(n)}{h(n)} \in \Theta(1)$.
2. If $f(n)$ is $\Theta(g(n))$, then $\log(f(n)) \in \Theta(\log(g(n)))$.