University of Waterloo
CS240 - Fall 2016
Assignment 1 Part 2

Due Date: Wednesday September 28 at 5:00pm

Please read `http://www.student.cs.uwaterloo.ca/~cs240/f16/guidelines.pdf` for guidelines on submission. Problems 5 – 8(a) are written problems; submit your solutions electronically as a PDF with file name `a01p2wp.pdf` using MarkUs. We will also accept individual question files named `a01q5w.pdf`, `a01q6w.pdf`, ... , `a01q8w.pdf` if you wish to submit questions as you complete them. Submit your solution to 8(b) electronically as a file named `report.cpp`.

Note: Assignment 1 has been split into 2 parts both worth 2.5% each. Keep in mind that part 1 is due before part 2. There are 39 marks for part 1 and 34 marks for part 2.

Problem 5 [2+2+4+4=12 marks]

Consider the following procedure.

```
pre: n is a positive integer
pre: v[1..n] is a binary vector of length n,
     i.e., each entry is either 0 or 1
foo(v,n)
1.   i := 1;
2.   while i<=n and v[i]=0 do
3        i := i+1
4    od;
5.   for j from 1 to i do
6.       print("Hello world!")
7.   od;
```

a) How many inputs are there are of size n?
Solution: There are totally 2^n inputs of size n.

b) What is the worst case number of calls to print? Give an exact formula in terms of n and justify your answer by giving an example of a worst case input of size n.
Solution: The worst case happens when all entries in v are 0, which takes n+1 iterations.

c) For $i \in \{1, 2, \ldots, n\}$, let S_i denote the subset of inputs of size n for which the number of calls to print is i. Describe and enumerate S_i.
Solution: Before enumerating S_i, I will give an simple example. Eg: v = [0, 0, 0, 0, 1, x, x, x, x], where v has size 9 and the first loop will terminate when i goes to

the fifth entry in the array, which is 1. Thus, $i = 5$ after the first loop. Therefore, the print is called 5 times totally. Because the first five entries is unchanged, the last four entries can be enumerated. Thus, the enumeration of S_i is $2^{9-5} = 2^4 = 16$.
Conclusively, the enumeration of $S_i = 2^{n-i}$.

d) What is the average case number of calls to print? Derive an exact closed form formula in terms of n.
Solution:

$T_A^{avg}(n) = \frac{1}{|I:size(I)=n|} \sum_{I:size(I)=n} T_A(I)$

$= \frac{1}{2^n} \sum_{i=1}^{n} i(2^{n-i})$

$= \sum_{i=1}^{n} \frac{i}{2^i}$

$= 2 - (n+2)/2^n$ from A1P4

Problem 6 [5 marks]

Prove that the following code fragment will always terminate.

```
s := 3*n  // n is an integer
while (s>0)
   if (s is even)
      s := floor(s/4)
   else
      s := 2*s
```

Solution:
If s is even, then it takes one operation to get one-fourth of its value. Because there is a $s > 0$ condition, if n is always a multiple of four, program will terminate in $\frac{1}{2}\log n + 4$ operations, where 4 means when n becomes 1, there are still four operations left, which is $s = 3 * 2 = 6 \rightarrow s = \frac{6}{4} = 1 \rightarrow s = 1 * 2 = 2 \rightarrow s = \frac{2}{4} = 0$, . Otherwise, if s is always odd, which is the worst case, there is one more operation in each iteration. Thus, it will take $floor(\log 3n)$ operations to complete the program. So, the program will terminate in both the best case and the worst case.

Problem 7 [5 marks]

Analyze the following piece of pseudo-code and give a tight bound (i.e. Θ notation) on the running time as a function of n.it Show your work. A formal proof is not required, but you should justify your answer.

```
1.  mystery ← 0
2.  for i ← 1 to 3n do
3.      mystery ← mystery × 4
4.      for j ← 1388 to 2010 do
5.          for k ← 4i to 6i do
6.              mystery ← mystery + k
```

Solution:
$\Theta(\sum_{i=1}^{3n} \sum_{j=1388}^{2010} 2i)$

$= \Theta(\sum_{i=1}^{3n} (2010 - 1388 + 1) * 2i)$

$= \Theta(\sum_{i=1}^{3n} 1246i)$

$= \Theta(1246 * \frac{(3n+1)*3n}{2})$

$= \Theta(n^2)$

Problem 8 [6+6=12 marks]

You are given an array $A[0 \ldots n-1]$ of integers (not necessarily distinct) that forms a max-heap of size n.

a) Describe an algorithm that takes as input an integer c, not necessarily in the heap, and reports all integers in the heap that are greater than or equal to c. The running time of your algorithm should be $O(k)$, where k is the number of integers reported. Provide a brief explanation for why the running time of your algorithm is $O(k)$.
Solution:
I will use the property of max-heap, and use recursion to find all nodes greater than integer c. If there is a function $findgreater(maxheap, c)$, the root needs to be checked first. If $root < c$, then return. Otherwise, prints the root, and do findgreater(maxheap→left, c) and findgreater(maxheap→right, c) at the same time. When either parts reach a node smaller than c, then recursion stops. Otherwise, prints all nodes greater than or equal to c.

b) Implement your algorithm from part (a). Your program should read from `cin` the size n, then the n integers in the heap $A[0 \ldots n-1]$, and finally the integer c, and then write to `cout` the integers in the heap that are greater than or equal to c. You may assume that every integer in the input is at least 0 and at most $2^{31}-1$ (so every integer will fit into a variable of type `int`).

Every integer in the input and output should be on a separate line. So for instance if the input consists of the following lines:

```
5
17
15
13
10
3
12
```

then your program should print out the integers `17`, `15`, and `13` in any order (and on separate lines).

Submit the code for your `main` function, along with any helper functions, in a file called `report.cpp`.