CS 577: Introduction to Algorithms Fall 2015

Homework 1

Instructor: Dieter van Melkebeek TAs: Kevin Kowalski, Andrew Morgan, Bryce Sandlund

This assignment covers program correctness.

General guidelines

- Read the problems, and start thinking about them right away. Plan your work in such a way that you have the opportunity to put some problems on the back burner for a while and revisit them later.
- Assignments consist of different types of problems:
  - *Review problems* are intended as preparation for the graded written problems, and are discussed during the review sessions. You should not turn them in.
  - *Graded written problems* are the only ones you have to turn in. You are required to work on them in groups of two to three, and turn in a single solution for all of you. Your solutions are due on paper at the start of the lecture one week after the problem is assigned. They are marked and graded, and model solutions will be provided.
  - *Additional written problems* are intended for further practice. You are strongly recommended to do those problems, but should not turn them in. You will receive model solutions for them.
  - *Optional programming problems* are coding problems which you can (but don't have to) submit to an on-line judge. You only get credit for a solution accepted by the online judge by December 15. In that case, you get full credit for the problem; otherwise, your grade will not be affected. Detailed instructions are posted on Moodle.
  - *Challenge problems* are intended for those who feel the assigned problems are too easy. You should not turn them in. We do not provide model solutions or hints for them.
- You are always supposed to substantiate the claims you make. In particular, whenever you are asked to design an algorithm, you should argue the correctness and the running time. You may use without proof any results covered in class, as long as you state and cite them properly.
- Model solutions are handed out at the end of the lecture of Section 2 on the day the assignment is due. Hence, no late assignments can be accepted. Your lowest homework score will be dropped at the end of the semester.
- Good luck!

Review problems

1. Consider the specification and procedure in Algorithm 1, where $n! = n \cdot (n-1) \cdot \ldots \cdot 2 \cdot 1$ is the factorial of n, with $0! = 1$. For example, $3! = 6$ and $4! = 24$.

Algorithm 1

Input: an integer $n \geq 0$
Output: $n!$

```
1: procedure FACTORIAL(n)
2:     r ← 1
3:     c ← n
4:     while c > 0 do
5:         r ← r · c
6:         c ← c − 1
7:     return r
```

   (a) State and prove an adequate loop invariant.

   (b) Use the loop invariant to prove partial correctness.

   (c) Specify and write an equivalent (tail) recursive program.

   (d) Prove correctness of your recursive program.

2. Consider the specification and procedure in Algorithm 2.

Algorithm 2

Input: an integer $n \geq 1$, an array $A[0..n-1]$ of integers that is *sorted* from smallest to largest, and an integer x
Output: -1 if x does not appear in A, otherwise the first index i such that $A[i] = x$

```
1: procedure BINARYSEARCH(n, A, x)
2:     i ← 0
3:     j ← n − 1
4:     while i < j do
5:         m ← ⌊(i + j)/2⌋
6:         if A[m] < x then
7:             i ← m + 1
8:         else
9:             j ← m
10:    if A[i] = x then
11:        return i
12:    else
13:        return −1
```

   (a) State and prove an adequate loop invariant.

   (b) Use the loop invariant to prove partial correctness.

   (c) Specify and write an equivalent recursive program.

Graded written problem

3. [10 points] Algorithm 3 contains four procedures that are supposed to compute $\gcd(a, b)$ for positive integers a and b. Explain for each procedure why it works or does not work. If the procedure does not work, provide an example input on which it fails; otherwise, give a correctness proof.

Algorithm 3 Four programs that are supposed to compute $\gcd(a, b)$ for positive integers a and b

```
1:  procedure GCD1(a, b)
2:      if a > b then
3:          a ← a − b
4:      else
5:          b ← b − a
6:      if a = b then
7:          return a
8:      else
9:          return GCD1(a, b)

10: procedure GCD2(a,b)
11:     if a = b then
12:         return a
13:     if a < b then
14:         return GCD2(b − a, b)
15:     else
16:         return GCD2(a, a − b)

17: procedure GCD3(a,b)
18:     if a = b then
19:         return a
20:     if a < b then
21:         return GCD3(a, b − a)
22:     else
23:         return GCD3(b, a · b)

24: procedure GCD4(a,b)
25:     if a = b then
26:         return a
27:     if a < b then
28:         return GCD4(a, b − a)
29:     else
30:         return GCD4(b, a · (b + 1))
```

Additional written problem

4. Consider the specification and procedure in Algorithm 4. This procedure is known as bubble sort. The algorithm consists of a number of phases that send a "bubble" from left to right, swapping two neighboring elements if they are out of order.

Algorithm 4 Sort an array of integers

Input: an integer $n \geq 0$ and an array $A[0..n-1]$ of integers
Output: array A sorted from smallest to largest

```
1:  procedure BubbleSort(n, A)
2:      m ← n
3:      while m > 1 do
4:          ℓ ← 0
5:          i ← 0
6:          while i < m − 1 do
7:              if A[i] > A[i + 1] then
8:                  swap A[i] and A[i + 1]
9:                  ℓ ← i + 1
10:             i ← i + 1
11:         m ← ℓ
```

(a) State and prove adequate loop invariants for the inner and outer while loops.
Hint: What can you say about the position of the largest element in A after one iteration of the outer while loop? What can you say about $A[m..(n-1)]$?

(b) Use the invariants to prove partial correctness and termination.

Optional programming problem

5. [2.5 points] Solve the SPOJ problem Life, the Universe, and Everything (problem code TEST) and submit it using the guidelines given in Moodle.

Challenge problem

6. Determine for each of the procedures from problem 3 the exact set of pairs (a, b) of positive integers for which the procedure returns $\gcd(a, b)$.