Mid-Term Exam

The Art of Computer Programming

17 November 2025

You have 1.5 hours to answer this mid-term exam. You are allowed 10 A4 sheets (i.e., 20 pages) with the content of your choice. No electronic devices or any other material is allowed. Do not forget to add your name on all your answer sheets and to number them. The exam is graded out of 20 points.

Exercise A: Program Analysis (4 points)

Consider the following Python function:

```
def process(lst):
    i = 0
    s = 0
    while i < len(lst):
        if lst[i] % 2 == 0:
            s += lst[i] // 2
            i += 1
        else:
            s += lst[i]
            lst.append(lst[i] - 1)
            i += 2
    return s
```

As a reminder, in Python, `//` and `%` are binary operators that compute, respectively, the quotient and the remainder of the integer (Euclidean) division of two positive integers (in this exercise, these operators are only used on positive integers).

Q1. (2 points) Consider the execution of `process([2,3,4])`. Give:

- at the beginning of each iteration of the `while` loop, the value of the variables `i` and `lst`;
- the final value returned by the function.

Q2. (1 point) Let n be the initial number of elements of `lst`. What is the maximum length the list `lst` may have at any point during the execution of the function? Justify your reasoning.

Q3. (1 point) Give an $O(\cdot)$ expression of the asymptotic worst-case time complexity of this function in terms of n. Justify your reasoning.

Exercise B: Pointer Manipulation (2 points)

Consider the following C code fragment:

```
int a = 3;
int b = 5;
int *p = &a;
int *q = p;
*p = b;
q = &b;
*q = *p + 2;
```

Q4. (2 points) What are the final values of `a`, `b`? (No explanation needed.)

Exercise C: Simple Algorithm Design (5 points)

Given a sequence $(a_0, a_1, \ldots, a_{n-1})$ of integers, a *prefix* of that sequence is a sequence $(a_0, a_1, \ldots, a_{k-1})$ such that $k \leqslant n$. We say that $(a_0, \ldots, a_{k-1})$ is a *strictly increasing* prefix if $a_0 < a_1 < \cdots < a_{k-1}$.

Design an algorithm (in pseudocode, Python, C, or C++) that takes as input a finite sequence of integers and returns the length of the longest prefix of that sequence that is strictly increasing. For instance:

- On input $(2, 5, 7, 3, 8)$, the algorithm should return 3.
- On input $()$, the algorithm should return 0.
- On input (10), the algorithm should return 1.
- On input $(4, 4, 5)$, the algorithm should return 1.

Q5. (2 points) Write your algorithm as a function, written either in clear pseudocode or as a Python, C, or C++ function.

Q6. (2 points) Argue that your algorithm is correct.

Q7. (1 point) Provide the asymptotic $O(\cdot)$ worst-case time complexity of this algorithm in terms of the length n of the input sequence. Justify your reasoning.

Exercise D: Asymptotic Complexity (4 points)

For each pair of functions below, determine whether $f(n)$ is in $O(g(n))$, whether $f(n)$ is in $\Omega(g(n))$, and whether $f(n)$ is in $\Theta(g(n))$. Justify briefly.

Q8. (1 point) $f(n) = n \log n + n, \quad g(n) = n\sqrt{n}$.

Q9. (1 point) $f(n) = n + 10, \quad g(n) = n \log n$.

Q10. (1 point) $f(n) = 3n^2 + 4n + 5, \quad g(n) = n^2$.

Q11. (1 point) $f(n) = n!, \quad g(n) = 3^n$. You can use the Stirling equivalent: $n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$.

Exercise E: Data Structures and OOP (5 points)

Consider the following C++ class implementing a simple multiset of integers (i.e., a collection to store integers in no specific order, where each integer may be stored multiple times).

```cpp
#include <vector>

class IntBag {
private:
  std::vector<int> data;

public:
  void add(int x) {
    data.push_back(x);
  }

  bool contains(int x) const {
    for (int y : data) {
      if (y == x) return true;
    }
    return false;
  }

  bool remove_one(int x) {
    for (unsigned i = 0; i < data.size(); ++i) {
      if (data[i] == x) {
        data.erase(data.begin() + i); // Remove element at position i of data
        return true;
      }
    }
    return false;
  }
};
```

Q12. (1 point) Consider the following code; give the final contents of the private `data` member of b after this code is executed. (No explanation needed.)

```
IntBag b;
b.add(3); b.add(5); b.add(3);
b.remove_one(3); b.add(5); b.remove_one(7);
```

Q13. (1 point) What is the asymptotic worst-case time complexity (in terms of the number n of stored elements) of the `remove_one` function? Justify briefly.

Q14. (1 point) Rewrite the method `contains` so that it returns the *number of occurrences* of `x` in the bag instead of a Boolean.

Q15. (2 points) Without writing it in detail, but just giving a high-level description of it in a few lines of text, propose an alternative way to implement the `class IntBag` that would have a lower overall complexity. Justify this complexity.