

# Exam (3/3rd term) Programming Fundamentals

Jan. 21 2026, 18:15-21:15h

- You can solve the problems in any order. Solutions must be submitted to the automated judgement system Themis.
- Grading: You can score a total of 100 points. You get 10 points for free. Exercise 1 is worth 30 points. The remaining exercises are worth 15 points each. Your grade will be the number of points divided by 10.
- For exercises 2, 3, and 4 you can score partial points if your program passes several but not all test cases. For example, if you pass 6 out of 10 test cases, then you are awarded 60% of the grade points for that exercise.
- Inefficient programs may be rejected by Themis. In such cases, the error will be ‘time limit exceeded’. The time limit for each problem is two seconds.
- The number of submissions to Themis is not limited. No points are subtracted for multiple submissions.
- There will be no assessment of programming style. However, accepted solutions are checked manually for cheating. For example, precomputed answers will not be accepted, even though Themis accepts them.
- Needless to say, you are not allowed to work together. If plagiarism is detected, both parties (supplier of the code and the person that sends in copied code) will be excluded from any further participation in the course.
- You are not allowed to use email, phones, smartwatches, tablets, calculators, etc. There is a calculator available on the exam computers (see icon on the desktop). You are allowed to consult the ANSI C book and a dictionary. You are not allowed to use a printed copy of the reader or the lecture slides, however they are available digitally (as a pdf) in Themis. You are allowed to access your own submissions previously made to Themis.
- For each problem, the first three test cases (input files) are available on Themis. These input files, and the corresponding output files, are called 1.in, 2.in, 3.in, 1.out, 2.out and 3.out. These files can be used to test whether the output of your program matches the requested layout, so that there can be no misunderstanding about the layout and spaces in the output.
- **If you fail to pass a problem for a specific test case, then you are advised not to lose much time on debugging your program, and continue with another problem. In the last hour of the exam, all test cases will be made visible in Themis.**

## Problem 1: Time complexity (30 points)

In this problem the specification constant  $N$  is a positive integer (i.e.  $N > 0$ ). Determine for each of the following program fragments the sharpest upper limit for the number of calculation steps that the fragment performs in terms of  $N$ . For a fragment that needs  $N$  steps, the correct answer is therefore  $\mathcal{O}(N)$  and not  $\mathcal{O}(N^2)$  as  $\mathcal{O}(N)$  is the sharpest upper limit.

```
1. int s = 0;
   for (int i=0; i < N; i+=3) {
       for (int j=3*N; j > 0; j/=2) {
           s += i*j;
       }
   }
```

- (a)  $\mathcal{O}(\log N)$  (b)  $\mathcal{O}(\sqrt{N})$  (c)  $\mathcal{O}(N)$  (d)  $\mathcal{O}(N \log N)$  (e)  $\mathcal{O}(N^2)$

```
2. int s = 0;
   for (int i=1; i < N*N; i*=3)
       s += i*i;
```

- (a)  $\mathcal{O}(\log N)$  (b)  $\mathcal{O}(\sqrt{N})$  (c)  $\mathcal{O}(N)$  (d)  $\mathcal{O}(N \log N)$  (e)  $\mathcal{O}(N^2)$

```
3. int i=0, s=0;
   while (s < N) {
       s += i;
       i++;
   }
   while (i > 0) {
       s += i;
       i--;
   }
```

- (a)  $\mathcal{O}(\log N)$  (b)  $\mathcal{O}(\sqrt{N})$  (c)  $\mathcal{O}(N)$  (d)  $\mathcal{O}(N \log N)$  (e)  $\mathcal{O}(N^2)$

```
4. int s=0;
   for (int i=0; i < N; i++) {
       for (int j=0; j < 5*i; j += 2) {
           s += i + j;
       }
   }
```

- (a)  $\mathcal{O}(\log N)$  (b)  $\mathcal{O}(\sqrt{N})$  (c)  $\mathcal{O}(N)$  (d)  $\mathcal{O}(N \log N)$  (e)  $\mathcal{O}(N^2)$

```
5. int s=0;
   for (int i=N; i > 0; i--) {
       int d = 2 + i % 5;
       for (int j=1; j < N; j *= d) {
           s += j*s;
       }
   }
```

- (a)  $\mathcal{O}(\log N)$  (b)  $\mathcal{O}(\sqrt{N})$  (c)  $\mathcal{O}(N)$  (d)  $\mathcal{O}(N \log N)$  (e)  $\mathcal{O}(N^2)$

```

6. int i=0, s=0;
   while (s < N*N) {
      s += i;
      i++;
   }

```

- (a)  $\mathcal{O}(\log N)$  (b)  $\mathcal{O}(\sqrt{N})$  (c)  $\mathcal{O}(N)$  (d)  $\mathcal{O}(N \log N)$  (e)  $\mathcal{O}(N^2)$

```

7. int a = 0, b = N*N;
   while (b - a > 1) {
      int c = (a + b)/2;
      a = (c*c > N ? a : c);
      b = (c*c > N ? c : b);
   }

```

- (a)  $\mathcal{O}(\log N)$  (b)  $\mathcal{O}(\sqrt{N})$  (c)  $\mathcal{O}(N)$  (d)  $\mathcal{O}(N \log N)$  (e)  $\mathcal{O}(N^2)$

```

8. int s=0;
   for (int i=1; i < N; i++) {
      for (int j=0; j < i; j += 10) {
         for (k=0; k < 10; k++) {
            s += i + j + k;
         }
      }
   }

```

- (a)  $\mathcal{O}(\log N)$  (b)  $\mathcal{O}(\sqrt{N})$  (c)  $\mathcal{O}(N)$  (d)  $\mathcal{O}(N \log N)$  (e)  $\mathcal{O}(N^2)$

```

9. int s = 0;
   for (int i=1; i < N; i++) {
      if (i*i > N) {
         break;
      }
      s += i;
   }

```

- (a)  $\mathcal{O}(\log N)$  (b)  $\mathcal{O}(\sqrt{N})$  (c)  $\mathcal{O}(N)$  (d)  $\mathcal{O}(N \log N)$  (e)  $\mathcal{O}(N^2)$

```

10. int s = 0;
    for (int i=0; i < N; i+=i) {
      s += i*i;
    }

```

- (a)  $\mathcal{O}(\log N)$  (b)  $\mathcal{O}(\sqrt{N})$  (c)  $\mathcal{O}(N)$  (d)  $\mathcal{O}(N \log N)$  (e)  $\mathcal{O}(N^2)$

```

11. int s=0, j = 1;
    for (int i = 1; i <= N; i++) {
      while (j*j <= i) {
         j++;
      }
    }

```

- (a)  $\mathcal{O}(\log N)$  (b)  $\mathcal{O}(\sqrt{N})$  (c)  $\mathcal{O}(N)$  (d)  $\mathcal{O}(N \log N)$  (e)  $\mathcal{O}(N^2)$

```

12. int s = 0, j = n;
    for (int i = 1; j > 0; i++) {
        j -= i;
        s += j;
    }

```

- (a)  $\mathcal{O}(\log N)$  (b)  $\mathcal{O}(\sqrt{N})$  (c)  $\mathcal{O}(N)$  (d)  $\mathcal{O}(N \log N)$  (e)  $\mathcal{O}(N^2)$

## Problem 2: Primonacci (15 points)

The function  $f(n)$  for integers  $n \geq 0$  is defined as follows:

$$f(n) = \begin{cases} n, & \text{if } n \leq 1, \\ f(n-1) + f(n-2), & \text{if } n \text{ is prime,} \\ 1 + f(n-1), & \text{otherwise.} \end{cases}$$

The input is a positive integer  $n$  (where  $0 \leq n \leq 10^6$ ). The program should output the remainder of dividing  $f(n)$  by 2026.

**Example 1:**

**input:**

0

**output:**

0

**Example 2:**

**input:**

5

**output:**

5

**Example 3:**

**input:**

6

**output:**

6

## Problem 3: Minimal Pair Distance (15 points)

Given an array  $a$  with  $n$  distinct non-negative integers, we define the *distance* between two elements  $a[i]$  and  $a[j]$  as the absolute value of their difference (in math notation:  $|a[i] - a[j]|$ ).

We define  $D$  to be the smallest possible distance between any two elements in the array. In other words,

$$D = \min_{0 \leq i < j < n} |a[i] - a[j]|.$$

The input for this problem consists of two lines. The first line contains an integer  $n$  (with  $n \geq 2$ ). The second line contains  $n$  distinct non-negative integers  $a[0], a[1], \dots, a[n-1]$ . The output must be two integers: the number  $D$ , followed by the number of unique pairs  $(a[i], a[j])$  that achieve the distance  $D$ . Note that the pair  $(a[i], a[j])$  and  $(a[j], a[i])$  are considered the same pair.

**Example 1:**

**input:**

2

0 42

**output:**

42 1

**Example 2:**

**input:**

5

1 4 7 11 20

**output:**

3 2

**Example 3:**

**input:**

6

1 4 7 10 13 16

**output:**

3 5

## Problem 4: Balanced Partition (15 points)

The input consists of two lines. The first line contains an integer  $n$  (where  $1 \leq n \leq 32$ ). The second line contains an array with length  $n$  containing digits (so  $0 \leq a[i] < 10$  for  $0 \leq i < n$ ).

The output of your program must be the number of ways in which the set of indices  $[0..n)$  can be split into two sets  $L$  and  $R$  such that:

- every index from  $[0..n)$  belongs to exactly one of  $L$  or  $R$ ,
- the sums of the digits in the two groups are equal:

$$\sum_{i \in L} a[i] = \sum_{i \in R} a[i].$$

Note that two partitions are considered equal if they only differ by swapping the sets  $L$  and  $R$ .

### Example 1:

#### input:

1

42

#### output:

0

### Example 2:

#### input:

4

1 1 1 1

#### output:

3

### Example 3:

#### input:

7

1 2 3 4 5 6 7

#### output:

4

## Problem 5: Dafny (15 points)

From Themis, you can download the file `problem5.dfy`. At several locations in the file, there are question marks. Replace the question marks by expressions, such that Dafny accepts the program fragment. Note that you are not allowed to add or remove extra statements, or change pre/post-conditions. You are only allowed to replace the question marks by suitable expressions. [Note: if you use the visual studio code environment, please make sure you save your file (Ctrl-s) before submitting it to Themis!]

```
method square(n: nat) returns (s: nat)
  ensures s == n*n
{
  s := ??;
  var i := ??;
  var odd := ??;
  while (??)
    invariant ???
  {
    s := s + ??;
    odd := odd + 2;
    i := i + 1;
  }
}
```