Exam Programming Fundamentals

January 22, 2025 (18:15-21:15)

- You can earn 90 points. You will get 10 points for free. So, you can obtain 100 points in total, and your exam grade is calculated by dividing your score by 10.
- This exam consists of 5 problems. The first problem consists of multiple choice questions. The problems 2, 3, and 4 are programming exercises in C. Problem 5 is about program correctness in Dafny. All problems are assessed by the Themis judging system. For each of the problems 2, 3, and 4, there are 10 test cases. Each of these test cases is worth 10% of the points.
- In the last hour of the exam, all inputs for the programming problems (2-4) will be made visible in Themis. Note that you can see the test cases of a problem only after having made a submission for that problem.
- Note that manual checking is performed after the exam. It is not allowed to use precomputed answers in your program. If this is detected by manual inspection, then all points for that exercise will be subtracted.
- This is an open book exam. You are allowed to use the ansi C book and the Dafny book. The pdf of the reader is available in Themis, as well as pdfs of the lecture slides. Any other documents are not allowed. You are allowed to use previous submissions that you made to Themis for the labs. Moreover, from Themis you can download an implementation of merge sort that you may use in your solutions.

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 $O(N)$ and not $O(N^2)$ as $O(N)$ is the sharpest upper limit.

1.
```
int i = 0, s = N*N;
while (s > 0) {
  i++;
  s -= i;
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

2.
```
int s = 0, i = N;
while (i > 0) {
  s += i;
  i = (i%2 ? i-1 : i/2);
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

3.
```
int bits=0;
for (int i=0; i < N; i++) {
  for (int j=i; j >= 0; j/=2) {
    bits += j%2;
  }
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

4.
```
int s = 0;
for (int i=1; i < N; i+=2) {
  s = s + N;
}
while (s > 0) {
  s--;
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

5.
```
int s = 0;
for (int i=2*N; i > 0; i--) {
  for (int j=1; j < N; j*=2) {
    s++;
  }
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

6.
```
for (int i=1; i < 10; i++) {
  int j = 0, k = 0;
  while (j < N) {
    j += 2*k + 1;
    k++;
  }
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

7.
```
int a = 0, b = N;
while (b - a > 1) {
  int c = (a + b)/2;
  a = (c*c > N ? a : c);
  b = (c*c > N ? c : b);
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

8.
```
int s = 0;
for (int i = 0; i < N; i +=2) {
  for (int j=1; j < i; j *= 2) {
    s++;
  }
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

9.
```
int d = 2, s = 0, n = N;
while (n > 1) {
  while (n%d == 0) {
    n = n/d;
    s++;
  }
  d++;
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

10.
```
int i = 0, j = N, k = 1;
while (i < j) {
  i += k;
  j++;
  k++;
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

11.
```
int s=0;
for (int i=1; i < 10; i++) {
  for (int j=i; j < N; j++) {
    for (int k=j; k < N; k++) {
      s += i + j + k;
    }
  }
}
```

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

12.
```
int s = 0, i = N;
while (i > 2) {
  s++;
  i = 1 + i/2;
}
```

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

Problem 2: Semiprime triplets (15 points)

A *semiprime* is a positive integer that is the product of exactly two prime numbers. The two primes in the product may equal each other. For example, the numbers $9 = 3 \times 3$, $35 = 7 \times 5$, and $33 = 3 \times 11$ are semiprimes.

A *semiprime triplet* is a triplet $(a, a+1, a+2)$ for which a, $a+1$, and $a+2$ are all semiprimes. An example is the triplet $(33, 34, 35) = (3 \times 11, 2 \times 17, 5 \times 7)$.

The input for this problem consists of an integer n (where $1 \leq n \leq 20000000$). The output must be the number of semiprime triplets $(a, a+1, a+2)$ with $a \leq n$.

Example 1:
input:
```
10
```
output:
```
0
```

Example 2:
input:
```
100
```
output:
```
3
```

Example 3:
input:
```
1000
```
output:
```
15
```

Problem 3: Counting pair sums (15 points)

Given an array of non-negative integers without duplicates, and a given positive integer n, you are requested to determine the number of sums $a + b$, where a and b are taken from the array and $a + b \geq n$. For an array with m numbers, we can construct $m(m-1)/2$ different sums $a + b$. Note that each number may be used only once (so $a \neq b$), and a sum like $x + y$ is considered the same sum as $y + x$.

The input of this problem consists of two lines. The first line contains the number n. The second line contains a positive integer `m` followed by a colon (`:`). Next follows an array of `m` non-negative `int` values. The output must be in the same format as the following examples.

Example 1:
input:
```
12
10: 1 2 3 4 5 6 7 8 9 0
```
output:
```
12
```

Example 2:
input:
```
12
5: 4 2 1 6 8
```
output:
```
2
```

Example 3:
input:
```
17
10: 16 11 7 14 18 13 8 12 1 4
```
output:
```
33
```

Problem 4: Interleaving arrays (15 points)

Given three arrays filled with integers, you are required to write a program that checks whether the third array is an interleaving of the other two arrays. An array `c[]` is called an *interleaving* of the two arrays `a[]` and `b[]` if it can be constructed by choosing integers from `a[]` and `b[]` while preserving the relative order of the integers from each array.

For example, if `a=[1,2,3]`, `b=[4,5,6]`, and `c=[1,2,4,3,5,6]` then `c` is an interleaving of `a` and `b`.

However, if `a=[3,2,1]`, `b=[4,5,6]`, and `c=[1,2,4,3,5,6]` then `c` is an not interleaving of `a` and `b`.

The input for this problem consists of three lines. The first line contains the array `a[]`, the second line the array `b[]`, and the third line contains the array `c[]`. Each array is specified on a line containing an integer n, which is the length of the array, followed by a colon (:) and n `int` values. You may assume that $1 \leq n \leq 50$. The output must be `YES` if the array `c[]` is an interleaving of the arrays `a[]` and `b[]`, and `NO` otherwise.

Example 1:

input:

```
3: 1 2 3
3: 4 5 6
6: 1 2 4 3 5 6
```

output:

```
YES
```

Example 2:

input:

```
3: 3 2 1
3: 4 5 6
6: 1 2 4 3 5 6
```

output:

```
NO
```

Example 3:

input:

```
3: 3 1 2
3: 1 5 6
6: 3 1 5 6 1 2
```

output:

```
YES
```

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 questions 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!]

```
function fac(n: nat): nat
  requires n >= 0
{
  if n == 0 then 1 else n*fac(n-1)
}

method factorial(n: nat) returns (f: nat)
  ensures f == fac(n)
{
  f := ??;
  var i := ??;
  while (??)
    invariant ??
  {
    var v,j := f,0;
    while (??)
      invariant ??
    {
      f := f + v;
      j:= j + 1;
    }
    i := i + 1;
  }
}
```