Exam Programming Fundamentals

January 24, 2024 (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 = 0;
while (s < N) {
  i++;
  s += 2*i + 1;
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

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

3.
```
int s = 1, t = 0;
while (s*s < N) {
  s = 2*s;
}
for (int i = 0; i < s; i++) {
  for (int j = 0; j < s; j++) {
    t += i + j;
  }
}
```
(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 j = N*N; j > 0; j /= 3) {
  for (int i = 0; i < N; i += 3) {
    s += i + j;
  }
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

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

6.
```
int s = 0;
for (int i = N*N; i > 0; i /= 2) {
  s = s + i;
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

7.
```
int s = 0, j = 0;
for (int i=0; i < N; i++) {
  j += i;
}
while (j > 0) {
  s += j;
  j -= 2;
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

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

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

10.
```
int s = 0, i = N;
while (i > 0) {
  s++;
  i = (i%2 ? i-1 : i/2);
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

11.
```
int s = 0, i = N;
while (i > 0) {
  i = (i%2==0 ? i/2 : i-1);
  for (int j=0; j < 2*N; j+=2) {
     s += i + j;
  }
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

12.
```
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) $O(N^2)$

Problem 2: Lychrel `int`s (15 points)

An integer is called *palindromic* if reversing its digits yields the same number. So, 12321 is palindromic, while 42 is not.

Let `n` be a non-negative `int`. We set `x=n`, and perform the following iterative process. If `x` is not a palindromic `int` then we reverse its digits and add the obtained number to `x` to get a new `x` and repeat this process. The process stops when `x` is a palindromic number, or when `x` becomes greater than $2^{31} - 1 = 2147483647$ (which is the maximal `int` value). In the latter case, we call the starting number `n` a *Lychrel* `int`.

For example, for `n=199` we find the following sequence of numbers (for `x`):

$$199 \rightarrow 199 + 991 = 1190 \rightarrow 1190 + 0911 = 2101 \rightarrow 2101 + 1012 = 3113$$

So, for `n`=199 the process ends with the palindromic number 3113, and we conclude that 199 is not a Lychrel `int`.

The input for this problem consists of an integer n (where $0 \leq n \leq 2^{31} - 1$). The output must be `YES` if n is a Lychrel `int`, otherwise it must be `NO`.

Example 1:
input:
```
199
```
output:
```
NO
```

Example 2:
input:
```
196
```
output:
```
YES
```

Example 3:
input:
```
42
```
output:
```
NO
```

Problem 3: Largest Contiguous Selection (15 points)

Consider an array containing the data `[2 3 0 1 3 1 4 0 1 8]`. From this array, we can construct several non-empty intervals $[a, b)$ of numbers that occur in the array. Recall that $[a, b)$ is notation for the set of integers x with $a \leq x < b$. For this example, these intervals are:

$$[0,1),\ [0,2),\ [0,3),\ [0,4),\ [0,5),\ [1,2),\ [1,3),\ [1,4),\ [1,5),\ [2,3),\ [2,4),\ [2,5),\ [3,4),\ [3,5),\ [4,5),\ [8,9)$$

Note that the interval $[0, 9)$ is not contained in the array because the numbers 5, 6, and 7 are missing.

As you can see, the array contains 5 numbers from the interval $[0, 2)$ (being twice a 0, three times an 1), while it contains 9 numbers from the interval $[0, 5)$ (being twice a 0, three times an 1, one 2, twice a 3, and one 4).

No other interval contains more than 9 numbers from the array. Therefore, this interval is printed on the output in the format `9: [0,5)`, meaning 9 numbers from the interval $[0, 5)$. If several intervals exist, with the same maximal number of elements from the array, then the interval with the smallest lower bound should be printed on the output.

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

Example 1:
input:
```
10: 0 1 1 1 1 2 9 10 10 11
```
output:
```
6: [0,3)
```

Example 2:
input:
```
10: 2 3 0 1 3 1 4 0 1 8
```
output:
```
9: [0,5)
```

Example 3:

input:

```
15: 13 5 14 17 5 2 0 17 18 6 1 3 16 15 2
```

output:

```
7: [13,19)
```

Problem 4: Maximal f-sum (15 points)

The integer 6 can be written as a sum of positive integers in 11 different ways:

$6 = 5{+}1 = 4{+}2 = 4{+}1{+}1 = 3{+}3 = 3{+}2{+}1 = 3{+}1{+}1{+}1 = 2{+}2{+}2 = 2{+}2{+}1{+}1 = 2{+}1{+}1{+}1{+}1 = 1{+}1{+}1{+}1{+}1{+}1$

Consider the array `[1 5 8 9 10 12]` which represents the function f such that $f(1) = 1$, $f(2) = 5$, $f(3) = 8$, $f(4) = 9$, $f(5) = 10$, and $f(6) = 12$.

We apply this function on the terms of the above sums as follows:

sum	f-sum
6	$f(6) = 12$
$5 + 1$	$f(5) + f(1) = 10 + 1 = 11$
$4 + 2$	$f(4) + f(2) = 9 + 5 = 14$
$4 + 1 + 1$	$f(4) + 2 \times f(1) = 9 + 2 \times 1 = 11$
$3 + 3$	$f(3) + f(3) = 2 \times 8 = \mathbf{16}$
$3 + 2 + 1$	$f(3) + f(2) + f(1) = 8 + 5 + 1 = 14$
$3 + 1 + 1 + 1$	$f(3) + 3 \times f(1) = 8 + 3 \times 1 = 11$
$2 + 2 + 2$	$3 \times f(2) = 3 \times 5 = 15$
$2 + 2 + 1 + 1$	$2 \times f(2) + 2 \times f(1) = 2 \times 5 + 2 \times 1 = 12$
$2 + 1 + 1 + 1 + 1$	$f(2) + 4 \times f(1) = 5 + 4 \times 1 = 9$
$1 + 1 + 1 + 1 + 1 + 1$	$6 \times f(1) = 6 \times 1 = 6$

The sums in the right hand side of the table are called f-sums. As you can see, the maximal f-sum for $n = 6$ is $f(3) + f(3) = 16$.

The input for this problem is a single line. The line starts with an integer n, which is the length of the array f, followed by a colon (:) and n `int` values. You may assume that $1 \leq n \leq 100$. The output must be the maximal f-sum that can be obtained for n.

Example 1:

input:

```
6: 1 5 8 9 10 12
```

output:

```
16
```

Example 2:

input:

```
8: 3 5 8 9 10 17 17 20
```

output:

```
24
```

Example 3:

input:

```
10: 7 2 3 9 5 6 5 2 7 5
```

output:

```
70
```

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!]

```
method problem5(a: array<int>) returns (k: nat)
requires a.Length > 1 && a[0] == a[a.Length-1]
ensures 0 <= k < a.Length - 1 && a[k] >= a[k+1]
{
  var i,j := 0, a.Length-1;
  while ??
    invariant ??
    decreases ??
  {
    var m := (i + j)/2;
    if ?? {
      i := ??;
    } else {
      j := ??;
    }
  }
  k := ??;
}
```