Exam Imperative Programming

duration: 3 hours

- 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 two problems are multiple choice questions. The problems 3, 4, and 5 are made using a computer. All problems are assessed by the Themis judging system. For each of the problems 3, 4 and 5, 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 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 pdf of the reader (which is available in Themis), pdfs of the lecture slides (also available in Themis), the prescribed ANSI C book (hard copy) and a dictionary. Any other documents are not allowed. You are allowed to use previous submissions that you made to Themis for the labs and the midterm(s).

Problem 1: Assignments (20 points)
For each of the following annotations determine which choice fits on the empty line (.....). The variables `x`, `y`, and `z` are of type `int`. Note that `A` and `B` (capital letters!) are specification constants (so not program variables).

1.1
```
/* x == A + 2 */
.....
/* x == 3*A + 1 */

(a) x = 3*x - 1;
(b) x = 3*(A - 1) + 4;
(c) x = 3*(x - 2) + 1;
```

1.2
```
/* 3*x + 8*y == A */
.....
/* 3*x + 5*y == A */

(a) x = x + y;
(b) x = x + 3*y;
(c) x = x - 3*y;
```

1.3
```
/* x + y == A, x + z == B */
x = x + y; y = y - z;
.....

(a) /* y == B, x + y == A */
(b) /* x == A, x - y == B */
(c) /* x == A, x + y == B */
```

1.4
```
/* x == A, y == B */
x = x - y; y = y - x;
.....

(a) /* x == A - B, y == 2*B - A */
(b) /* x == B - A, y == 2*A - B */
(c) /* x == A - B, y == A - 2*B */
```

1.5
```
/* x == B, y == A */
x = x - y; y = x + y; x = y - x;
.....

(a) /* x == B, y == A */
(b) /* x == A, y == B */
(c) /* x == A, y == A */
```

1.6
```
/* x == A, y == B */
y = x - y; x = x + y; y = x + y;
.....

(a) /* x == A - B, y == 2*A - B */
(b) /* x == 3*A - 2*B, y == A - B */
(c) /* x == 2*A - B, y == 3*A - 2*B */
```

Problem 2: Time complexity (20 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 n=1, d = 1;
while (d < N) {
  d = d + 2*n + 1;
  n++;
}
```
(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; j > i; j-=2) {
    s++;
  }
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

3.
```
int s = 0, i = 0;
while (i < N) {
  i += (s%2 == 0 ? 1 : 2);
  s++;
}
```
(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 = 0; i < N; i++) {
  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) $O(N^2)$

5.
```
int l = 0, r = N;
while (l + 1 < r) {
  int m = (l + r)/2;
  if (N < m*m) {
    r = m;
  } else {
    l = m;
  }
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

6.
```
int a=0, b = 1, n = 0;
while (n < N) {
  b = a + b;
  a = b - a;
  n++;
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

Problem 3: Goldbach (15 points)

Goldbach's conjecture is one of the oldest unsolved (i.e. not proven) problems in mathematics. It states that every *even* natural number greater than 2 can be written as the sum of two prime numbers. The conjecture has been shown to hold (using computer verification) for all integers less than 10^{18}.

The input for this problem is an even integer n, where $4 \leq n \leq 6000000 = 6 \times 10^6$. The output must be the number of pairs (a, b) such that $a + b = n$ (where $a < b$) and a and b are both prime numbers.

Example 1:
input:
```
16
```
output:
```
2
```

Example 2:
input:
```
42
```
output:
```
4
```

Example 3:
input:
```
256
```
output:
```
8
```

Problem 4: Intervals (15 points)

The input for this problem consists of two lines. The first line contains a positive integer n, which is the number of items in the array of integers on the second line. This array should be regarded as a set, so the order and duplicates are irrelevant. For example, the sets $\{1, 2, 3\}$, $\{3, 1, 2\}$ and $\{1, 1, 2, 3, 2\}$ are all the very same set.

A set like $\{1, 4, 6, 8, 10, 11, 12, 2, 13, 14, 3, 15, 9, 9, 1\}$ can be represented more concise as `1..4,6,8..15`, meaning that the numbers from the intervals [1..4] and [8..15] are in the set, as well as the single element 6. Note that the intervals are maximal in size, and increasingly ordered on their smallest value (so, `8..9,15,9..14,1..4,6` is not valid). Make a program that reads the input (as described above) and outputs the concise set representation of the input.

Example 1:
input:
```
7
1 3 5 2 8 0 9
```
output:
```
0..3,5,8..9
```

Example 2:
input:
```
10
2 4 0 1 2 0 1 0 2 1
```
output:
```
0..2,4
```

Example 3:
input:
```
10
5 6 4 4 0 3 2 6 2 2
```
output:
```
0,2..6
```

Problem 5: Sum with k terms (20 points)

The input for this problem consists of two lines. The first line contains three integers s, k, and n where $2 \leq k \leq n \leq 25$. The second line contains an array with n integers.

The output of your program must be `YES` if there are k elements in the array that have a sum that equals s. It is not allowed to use the same array element more than once in this sum. Note that this does not mean that the sum consists of unique terms because an array may contain duplicate numbers.

Example 1:
input:
```
20 4 8
1 3 5 4 0 9 2 7
```
output:
```
YES
```

Example 2:
input:
```
26 4 8
1 3 5 4 0 9 2 7
```
output:
```
NO
```

Example 3:
input:
```
42 5 10
1 3 5 7 12 4 11 9 21 7
```
output:
```
YES
```