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 precomuted 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`, and `y` are of type `int`. Note that `A` and `B` (uppercase letters!) are specification constants (so not program variables).

```
1.1  /* x == A */
     .....
     /* x == 3*A + 18 */

    (a) x=3*x + 18;
    (b) x=x/3 - 18;
    (c) x=(x - 18)/3;
```

```
1.2 /* 4*x + 2*y + 2*z == 2*A */
    .....
    /* x == A */

    (a) x=x/2 + y; x=y/2 + z;
    (b) x=2*x - y; x=x + 2*y + z;
    (c) x=2*x + z; x=x - y;
```

```
1.3 /* 3 <= x + y * y < 12 */
    .....
    /* 5 <= y < 14 */

    (a) y=x + y * y + 2;
    (b) y=x + y * y - 2;
    (c) y=y * y; x=2;
```

```
1.4 /* x == A, y == B */
    x=2*x - y; y=y + x; x=x - y;
    .....

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

```
1.5 /* 4*x + 2*y + 2*z < 12 */
    y=y + z; x=2*x + y;
    .....

    (a) /* 8*x < 6 */
    (b) /* x < 6 */
    (c) /* x < 12 */
```

```
1.6 /* x == A, y == B */
    x=2*x + 2*y; y=x - 2*y; x=(x-y)/2;
    .....

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

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

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

3.
```
int s = 0;
for (int i = 0; i < N; i++) {
  for (int j = 5; j > 0; j--) {
    s += j;
  }
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

4.
```
for (int i = 0; i < N; i++) {
  int j = i;
  while (j > 0) {
    j /= 2;
  }
}
```
(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 = 1; i < N * N; i = i * 3) {
  s++;
}
```
(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; i > N / 2; i = i - 2) {
  s = s + i;
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

Problem 3: Number Permutations (15 points)
A positive integer n is a *permutation* of a positive integer m if its digits can be re-arranged such that m is obtained. For example, 123 is a permutation of the numbers 123, 132, 213, 231, 312 and 321.

The input for this problem consists of two integers n and m (where $1 \leq n, m \leq 10^9$). The output must be `YES` if n is a permutation of m, otherwise it must be `NO`.

Example 1:
input:
```
123 321
```
output:
```
YES
```

Example 2:
input:
```
6 60
```
output:
```
NO
```

Example 3:
input:
```
1001 110
```
output:
```
NO
```

Problem 4: Minimal Number Of Deletions (15 points)
The input for this problem are two arrays of `int`s. The output must be the minimum number of deletions from either array such that they become a permutation (shuffling) of each other.

For example, consider two arrays containing the data `[1,2,2,3,4,5,8]` and `[2,2,3,5,1,7]`. If we delete the numbers `4` and `8` from the first array, and the number `7` from the second array, then the remaining arrays are `[1,2,2,3,5]` and `[2,2,3,5,1]` which are permutations of each other. Hence, 3 deletions are needed.

The input of this problem consists of two lines, containing the two arrays. The first number n on a line is the length of the array, followed by a colon (:) and n `int` values.

Example 1:
input:
```
7: 1 2 2 3 4 5 8
6: 2 2 3 5 1 7
```
output:
```
3
```

Example 2:
input:
```
5: 1 0 0 0 1
7: 0 1 1 1 1 1 0
```
output:
```
4
```

Example 3:
input:
```
10: 2 1 0 1 1 2 2 4 3 3
10: 2 1 1 3 4 2 1 3 2 0
```
output:
```
0
```

Problem 5: Choice Maximization (20 points)
In this problem you are given a series of positive `int`s. The output must be the maximal sum that can be obtained by choosing numbers from this series under the constraint that you are not allowed to choose three or more consecutive elements.
For example, for the series `[5,4,2,3,1,7]` this maximum sum would be `5+4+3+7=19`. The sum `5+4+3+1+7=20` is invalid because it uses three numbers in a row (being `3`, `1`, and `7`).
The input for this problem consist of a single line. The line starts with a number n which is the length of the array, followed by a colon (:) and n `int` values. You may assume that $1 \leq n \leq 30$. The output must be the maximal sum that can be obtained.

Example 1:
input:
```
6: 5 4 2 3 1 7
```
output:
```
19
```

Example 2:
input:
```
10: 4 7 8 6 3 7 5 3 7 5
```
output:
```
42
```

Example 3:
input:
```
10: 5 8 2 8 4 8 3 3 7 1
```
output:
```
40
```

This sheet can be used as scratch paper.

This sheet can be used as scratch paper.