Michiel Teunar

397 109 307 184 440 143 221 153 28 207

397

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

1.1
```
/* x' == y'*z' */
.....
/* x == y*z' */
```
(a) `y--; x = x - z;`
(b) `z--; x = x + y;`
(c) `y++; z--;`

1.2
```
/* 3*x + 5*y == A */
.....
/* y == A */
```
(a) `y = (A - 3*x)/5;`
(b) `x = x + y; y = 3*x + 2*y;`
(c) `y = (A - 3*x)/5;`

1.3
```
/* x*x <= A < (x+1)*(x+1) */
.....
/* x <= A < x + y */
```
(a) `y = 2*x + 1; x = x*x;`
(b) `x = x*x; y = 2*x + 1;`
(c) `y = 2*x - 1; x++;`

1.4
```
/* x == A, y == B */
x = x - y; y = x + y; x = 2*x - y;
```
(a) `/* x == A - 2*B, y == A */`
(b) `/* 2*x == B, 2*y == A - B */`
(c) `/* x == 3*A - 2*B, y == A + B */`

1.5
```
/* x == A, y == B */
x = 3*x + y; y = x - y; x = x - y;
```
(a) `/* x == B, y == A */`
(b) `/* x == B, y == 3*A */`
(c) `/* x == 3*B, y == A */`

1.6
```
/* x == A, y == B */
x = 2*x + 2*y; y = 2*x - y; x = y - x;
```
(a) `/* 2*y - 4*x == A, y - x == B */`
(b) `/* x == 4*B - 6*A, y == B - 2*A */`
(c) `/* x == 2*A + B, y == 4*A + 3*B */`

Problem 2: 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 d = 2;
while (d*d <= N) {
  if (n%d == 0) {
    break;
  }
  d++;
}
```
(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++) {
  s += i;
}
for (int i = s; i > 1; i -= 2) {
  s++;
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

3.
```
int d = 2, n = N;
while (n > 1) {
  n = (n%d == 0 ? n/d : n);
  d++;
}
```
(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 a = 42, n = N, p = 1;
while (n > 0) {
  if (n%2 == 1) {
    p *= a;
  }
  a = a*a;
  n /= 2;
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

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

30 minutes. ✓

Problem 3: Longest Subsequence (15 points)
The input of this problem is a series of lower case letter from the alphabet (i.e. 'a'..'z'). For each letter that occurs in the input, the output must show the length of the longest consecutive subsequence containing only that letter. Take for example the sequence `abbaababb` consisting of `a`'s and `b`'s. In this sequence, `aa` and `bb` are the longest consecutive subsequences of repeating characters, each having length 2. Note that a single character is a sequence that has length 1.

The input of this problems consists of a string containing lower case letters. On the output, for each occurring letter in the input, you should print the letter followed by a colon (':') and the length of the longest subsequence for that letter (see examples below). Note that the output must be in alphabetical order.

Example 1:
input:
`abbaababb`
output:
`a:2`
`b:2`

Example 2:
input:
`aaabb`
output:
`a:3`
`b:2`

Example 3:
input:
`abzzccczzba`
output:
`a:1`
`b:1`
`c:3`
`z:2`

lowercase letters → a → 100 } → char array of 22
z → 122 }

Problem 4: Maximum Sum (15 points)
The input for this problem consists of two lines. The first line contains an integer n, where $3 \leq n \leq 35$. The next line contains n positive integers. The output should be the maximum sum of non-adjacent values from the input. Take for example the sequence `[3, 4, 6, 1, 2, 3]`. All sums without non-adjacent values are:

$3+6+2 = 11$
$3+6+3 = 12$
$3+6 = 9$
$3+1+3 = 7$
$3+1 = 4$
$3+2 = 5$
$3+3 = 6$
$4+1+3 = 8$
$4+1 = 5$
$4+2 = 6$
$4+3 = 7$
$6+2 = 8$
$6+3 = 9$
$1+3 = 4$

The maximum sum is 12, which should be the output for this example.
Note that the sum $4 + 6 + 3 = 13$ is greater than 12, but it uses the two adjacent values 4 and 6, which is not allowed. Also note, which is clear from the above example, that in this problem a sum must have at least two terms.

Example 1:
input:
6
3 4 6 1 2 3
output:
12

Example 2:
input:
3
40 41 2
output:
42

Example 3:
input:
3
27 81 55
output:
82

30 min ✓

Problem 5: Longest Palindromic Sequence (20 points)
The input for this problem consists of two lines. The first line holds a single positive integer n. The second line contains a string containing n lower-case letters from the alphabet (i.e. 'a'..'z').

The output should be the length of the longest palindromic subsequence that can be obtained by removing zero or more letters from the input string.

Take for example the word `characters`. Removing the characters `h`, `t`, `e`, `r` and `s` yields the longest possible palindromic subsequence `carac` which has length 5.

Example 1:
input:
10
characters
output:
5

Example 2:
input:
9
recursion
output:
3

Example 3:
input:
10 123
abbaisback
output:
5 7