Exam Imperative Programming

Friday, December 7th, 2018, 19:00 h.

- 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 test case is worth 10% of the points.
- Note that manual checking is performed after the exam. For example, if a recursive solution is requested then a correct iterative solution will be rejected after manual checking, even though Themis accepted it. Also, precomputed answers will be rejected.
- 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.
- You are allowed to take the exam text home.

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 + 7 */
    .....
    /* x == 11 */

    (a) y = 4;
    (b) x = x - y + 4;
    (c) x = 11 - (x + y + 7);
```

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

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

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

    (a) x = x + y; y = y + x;
    (b) x = x + 1; y = y + 1;
    (c) x = y; y = x;
```

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

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

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

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

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

    (a) /* x == 2*A, y == 3*B - A */
    (b) /* x == 2*B, y == B - A */
    (c) /* x == 2*A + 2*B, y == 3*B - 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 = N, s = 0;
   while (i > 1) {
     i = i/2;
     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;
   for (int i = 0; i < N; i+=10) {
     s += i;
   }
   ```
   (a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

3. ```
   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) $O(N^2)$

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

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

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

Problem 3: Ordered Pairs (15 points)
The input of this problem consists of two lines. The first line contains a positive integer n, and the second line consists of an array a of n non-negative integers. The output of the program must be the number of ordered pairs (i, j) such that $i < j$ and $a[i] > a[j]$.

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

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

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

Problem 4: Maximal sum (15 points)
The input of this problem consists of a single line containing a series of non-zero integer values that is terminated by a zero. At least one of the numbers is positive. Given this series of numbers, the output of your progam must be the largest possible sum of a consecutive non-empty subseries. Note that the entire series is considered to be a subseries as well. You may assume that the input numbers are chosen such that the maximum sum will not overflow.

For example, consider the input series `-5 1 3 -3 6 -2 4 -1 0`. The consecutive subseries `1 3 -3 6 -2 4` has the maximal sum, being 9.

Example 1:
input:
```
-5  1 3 -3 6 -2 4 -1 0
```
output:
```
9
```

Example 2:
input:
```
1 4 -2 6 0
```
output:
```
9
```

Example 3:
input:
```
-2 1 4 -5 6 -1 2 0
```
output:
```
7
```

Problem 5: Minimal Palindromic Partition (20 points)
The input for this problem is a string that is at most 20 characters long. Its characters are uppercase letters from the alphabet (i.e. 'A'..'Z'). A partitioning of the string in substrings is called a *palindromic partitioning* if every substring of the partition is a palindrome. For example, the string `"MINIMALLY"` can be partitioned in 4 palindromes: `"MINIM"`, `"A"`, `"LL"`, and `"Y"`. Of course, there exists other palindromic partitionings of this string. For example, we can also partition the word in 6 palindromes: `"M"`, `"INI"`, `"M"`, `"A"`, `"LL"`, and `"Y"`.

The output of your program must be the number of palindromes in the palindromic partitioning of the input string that has the fewest number of palindromes. So, for the string `"MINIMALLY"`, the output should be `4`.

The following incomplete code fragment is available from Themis (file `palpartition.c`). You must implement the body of the function `minimalPalPartition`. It must be a *recursive* function, or it should call a *recursive* helper function with suitably chosen parameters/arguments. You are not allowed to make changes in the `main` function.

```
int minimalPalPartition(int i, int j, char *a) {
  /* Implement the body of this function.
   * Moreover, this function must be recursive, or
   * it should call a recursive helper function.
   */
}

int main(int argc, char *argv[]) {
  char a[21];
  scanf("%s", a);
  printf("%d\n", minimalPalPartition(0, strlen(a)-1, a));
  return 0;
}
```

Example 1:
input:
```
MINIMALLY
```
output:
```
4
```

Example 2:
input:
```
RACECAR
```
output:
```
1
```

Example 3:
input:
```
LAVALETTE
```
output:
```
2
```