Algorithms and Data structures 2017: Second assignment

1 Instructions

You may work in groups of two. For this assignment you have to hand in two things:

- Source code. We will run the code for grading.
- A report. In the report you have to explain the algorithm and analyse it.

Source code You are allowed to submit a solution in either C, C++, Java or Python. If you prefer another language, please contact us. You are only allowed to use the Standard Library corresponding to your selected language.
You will find a set of examples to test your solution on the course wiki. We will set up a website, on which you can upload your code. It will be automatically be tested against test cases.

Report Besides handing in code, we would like to receive a report in which you explain your algorithm and analyse its correctness and runtime complexity.

Submission The deadline for sending in your solution is on January 11, 23:59 CET. You must submit your solutions via Blackboard (both the code and the report and nothing more). Only one team member has to submit a solution; the names and student numbers of both team members must be mentioned in the report.

Grading Grades will be determined as follows. You may earn up to 100 points for your solution:

- 15 points for the explanation of your algorithm.
- 10 points for the correctness analysis.
- 10 points for the complexity analysis.
- 50 points for the test results. We will be running several tests and you will get points for every correct answer within the time limit. If your code does not compile or does not read and write via `stdin` and `stdout`, you will get zero points on the test cases. So please test your code well!
- 15 points for the quality of the code.

If you have any questions, do not hesitate to send a email to Joshua (or someone else related to the course).

2 Merry and Pippin go shopping

Merry and Pippin are two close friends which used to be crooks. They stole vegetables from farmer Maggot and did all kinds of other mischief. To better their lives the swore to never steal something again. But unfortunately this means they have to pay for their vegetables now. They are now always looking for ways to "hack the system" and pay less.

Figure 1: Merry and Pippin stealing veggies

The two friends heard that Dutch people are a bit like them, as Dutch people always try to buy things in discount or try to haggle. A typical trick of the Dutch is the following. As you know, when you pay by cash in a Dutch supermarket, the total sum is rounded to multiples of 5 cents. (For example, when you buy a single bell pepper you have to pay 90 cents instead of the 89 cents it actually costs. On the other hand, when you buy one bag of carrots, normally costing 76 cents, you only have to pay 75 cents.) Dutch people exploit this rounding systems by reordering their products and grouping them cleverly with the checkout dividers (beurtbalkjes). Doing this properly, each group of products will be rounded down, saving some cents.

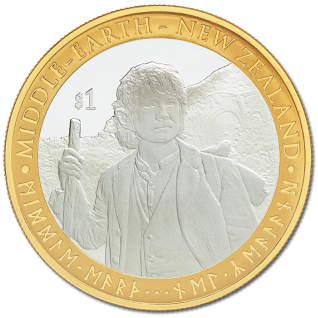

Figure 2: One dollar

A similar rounding system is used in shops in New Zealand, where Merry and Pippin obviously live. However, they round to the nearest multiple of 10 cents instead of 5. This means that the values $0, \ldots, 4$ are rounded down to 0 cents, and the values $5, \ldots, 9$ are rounded up to 10 cents.

Merry and Pippin already put their vegetables on the conveyor belt at the cashier. They want to know where to put the checkout dividers in order to save some precious cents. (They are too lazy to reorder the products.) So can you solve the following problem for them? Given n products in order with costs $c_1, \ldots, c_n$ and given k dividers, can you place the dividers in such a way to total amount they have to pay is minimised?

2.1 Input

You are given the following data (via `stdin`):

- One line with the number of products: n ($1 \leq n \leq 5000$) and the number of dividers: k ($0 \leq k \leq 50$).
- One line with all the n costs in cents: $c_1, \ldots, c_n$ ($1 \leq c_i \leq 100000$). The products come in order they are put on the conveyor belt, i.e., c_1 is closest to the cashier.

2.2 Output

You should return (via `stdout`):

- One line with the minimal amount (in cents) Merry and Pippin have to pay to buy all the products, using at most k dividers.

We will run your program and check exactly with this specification. Any other output (besides whitespace) will be regarded as a wrong output.

Example input 1:

```
8 1
1 1 1 1 1 1 1 1
```

Output:

```
0
```

Example input 2:

```
5 2
10 17 17 17 37
```

Output:

```
80
```

Example input 3:

```
7 2
25 26 11 15 16 12 4
```

Output:

```
100
```

Example input 4:

```
4 1
25 16 21 24
```

Output:

```
80
```