Algorithms and Programming
13 September 2018
Part II: Program (12 point version)

At most one C manual is allowed. Examination time: 100 minutes. Final program due by 11.00 p.m. of Monday the 17th; use the course portal page ("Elaborati" section) to up-load it.

1 (2.0 points)
In a football tournament r teams play c times against each other. All tournament results are stored in a matrix `m` with `r` rows and `c` columns. Each column of the matrix represents results of all r teams in one day of the tournament. The matrix includes integer values which indicate game results: 3 the team wins, 1 is a draw, and 0 the team loses. The winning team gets three points, the losing team does not get any points, and the teams get one point each for a draw.
Write function

```
void ranking_write (int **m, int r, int n);
```

which writes (on standard output) the leader team and its number of points after each day of the tournament.
For example, if $r = 4$, $c = 3$, and `m` is the following one:

3	1	0
0	1	1
1	1	1
1	1	3

The leader team is team 0 (with 3 points) after the first day, team 0 (with 4 points) after the second day, and team 3 (with 5 points) after the third day.

2 (4.0 points)
A list, with elements of type `list1_t`, stores integer values. Write a function able to "compress" such a list, i.e., to create a new list, with elements of type `list2_t`, in which each element stores an element of the first list and the number of times such an element appears in the list itself.
The function prototype must be

```
list2_t *list_compress (list1_t *p1);
```

where `p1` is a pointer to the head of the first list, and the function returns the pointer to the second list.
For example, let us suppose that `p1` points to the list $\{3, 3, 3, 3, 4, 4, 2, 6, 3, 6, 4\}$. The function has to create the following list $\{(3, 5), (4, 3), (2, 1), (6, 2)\}$ and to return its head pointer.
The candidate must also define both node structures.

3 (6.0 points)
An array named `cake` contains `n` elements. Each element is a C structure including 3 non-negative integers:

- The first one, named `code` specifies the cake code, e.g., 1 for muffins, 2 for scones, 3 for a carrot pastries, etc.
- The second one, named `quantity` indicates the number of cakes available, e.g., 4, 10, etc.
- The third one, named `weight` specifies the weight of the cake in grams, e.g., 100, 140, etc.

Write function

```
void basket_generate (array_t *v, int n, int w);
```

which receives a pointer to the specified array `v`, its size `n`, and an integer `w`. The function must generate a basket of pastries whose total weight is the closest possible one to `w`, keeping into account the pastries availability and their weight. In case there exist more than one equivalent solution, the program has to print only one of them.

Algorithms and Programming
13 September 2018
Part II: Program (18 point version)

At most one C manual is allowed. Examination time: 100 minutes. Final program due by 11.00 p.m. of Monday the 17th; use the course portal page ("Elaborati" section) to up-load it.

It is required to write an application which manipulates transactions on all world-wide stock markets. Each stock market regularly sends to the application a file storing all transactions made during the day. Each transaction is stored on a file line, and it includes the following pieces of information (space separated):

- The share name, reported as an alphanumeric string of at most 50 characters, e.g., Amazon, Intel, BWM, etc.
- The transaction date, in the format *dd.mm.yyyy*, e.g., 13.02.2018, 10.02.2018, etc.
- The transaction time, in 24-hour format (*hh.mm*) and referred to Greenwich time (GMT), e.g., 10.00, 16.33, etc.
- The share value as a real and positive value in USA dollars, e.g., 12.45, 234.33, etc.

The number of file lines is unknown as it is the order of the transactions.
Write a C program able to accept the following commands:

- `read fileName`
  It reads a new file, with name `fileName`, and it stores or adds all information in a proper data structure. Notice that, reading operations must be incremental, i.e., previously stored information items must be preserved when a new reading operation is performed. Moreover, the date and the time **must** be stored in an single C data structure with separate integer fields to store year, month, day, hours and minutes. Furthermore, the candidate **must** write proper functions to store, retrieve, and compare dates and times in such a data structure.
- `title shareName`
  It writes on standard output all transactions for the share with name `shareName`. This operation must be performed with an average cost that must be logarithmic or lower in the number of shares stored in the data structure.
- `interval shareName` $\texttt{date}_1$ $\texttt{time}_1$ $\texttt{date}_2$ $\texttt{time}_2$
  It writes on standard output the minimum and maximum values of all transactions recorded for the share with name `shareName` within the specified interval of time, i.e., going from $\texttt{date}_1$ $\texttt{time}_1$ to $\texttt{date}_2$ $\texttt{time}_2$ (range boundaries included).

  Notice that this operation can be logically divided into two steps. In the first one, the share with the given name must be found. Then, the maximum and minimum values must be found within all transactions recorded for that share. The first step must be performed with the same cost of the previous command. The second one must be performed with an average cost that is logarithmic or lower in the number of transactions stored for that share.