Algorithms and Programming (01OGD)
26 January 2016
Part II: Program (18 point version)

No books or notes are allowed. Examination time: 100 minutes.
Final program due by Friday the 29th, 12.00 p.m.; send the program by e-mail to `stefano.quer@polito.it`.

N cities are placed along a straight road. Each of them is characterized by a name, the number of citizens, and its distance from the origin of the straight road. The distance between any two cities is given by the difference of their distances with respect to the origin of the road.
All city data are stored in a file, of which the following one is a correct example:

```
11
Piacenza 102 0
Fidenza 27 35
Parma 190 57
ReggioEmilia 171 83
Modena 185 115
Bologna 386 144
Imola 69 177
Faenza 58 193
Forl 118 207
Cesena 97 225
Rimini 147 253
```

The first row of the file specifies the number of cities. For each city the file stores the name (C string of maximum 20 characters), the number of citizens (in thousands), and the distance from the origin of the straight road (in kilometers).
The government wants to place k ministries of the government on those N cities, such that, roughly speaking, the ministries are as closed as possible to the largest possible number of citizens. Mathematically speaking, the ministries have to be placed in cities such that the following function is minimized:

$$\sum_{i=0}^{N-1} \left[(citizens_{city_i}) \cdot (distance_{(city_i - city_{closest\ ministry})}) \right]$$

where for each city i (out of N cities) the function considers the product between the number of citizens of the city itself, and the distance from the city to the closest city in which a ministry has been placed (the distance is equal to zero if the city itself has a ministry).

Write a C program able to:

- Read from the command line the name of the file storing all city data, and the number of ministries k to place in those cities.
- Store all city information in a proper data structure.
- **Find in which cities to place all k ministries to minimize the previously reported cost function**. As such a cost function can be implemented by a proper C function running in $O(N)$ time, the function implementation will be subject to evaluation.

Explicitly state on the first page of the solution which "Combinatoric principle" has been used to solve the problem, and motivate (i.e., $2-3$ lines of English text) why this choice has been made.

Algorithms and Programming (01OGD)
26 January 2016
Part II: Program (12 point version)

No books or notes are allowed. Examination time: 100 minutes.
Final program due by Friday the 29th, 12.00 p.m.; send the program by e-mail to `stefano.quer@polito.it`.

1 (2.0 points)
Write function

```
char *charErase (char *str, int *pos);
```

which receives a string `str` and an array of integer values `pos`, and it returns the string generated by `str` erasing all characters in the position indicated by the array `pos`. The returned string must be allocated. The array `pos` has unknown size, but its last element has value `-1`.
For example, if the string `str` is the following one:

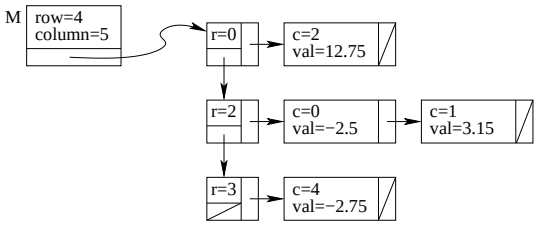
0	1	2	3	4	5	6	7	8	9	10	11	12	13
T	h	i	s	I	s	A	S	t	r	i	n	g	\0

and the array `pos` the following one:

7	4	2	0	11	-1

It is necessary to erase characters `S`, `I`, `i`, `T`, `n`, from the string `str`, and return string `hssAtrig`.

2 (4.0 points)
A sparse matrix, i.e., a matrix in which only non-zero elements are memorized, can be stored as as a list of lists. The main list includes an element for each row for which there is at least one element which differ from zero. Elements of that list store the row index. Each secondary list include only elements of that row which differ from zero. Each element of those lists store the column index of the element, and the element value. The following is an example of such a matrix (with 4 elements stored):



Write function

```
void matWrite (mat_t *M, float value, int r, int c);
```

which write the real value named `value` in the element of row `r` and column `c` of the matrix. If the element of position (`r`, `c`) already exists, its value has to be updated.
Indicate the data type used to implement the sparse matrix. Notice that `M` is an handle storing the size of the matrix (number of rows and columns) and the pointer to the first element of the main list.

3 (6.0 points)
Write function

```
void stringSplit (char *str, int n, int *length, ...);
```

which is able to split the string `str` into an undefined number of sub-strings whose sizes are stored in `length[0]`, `length[1]`, ..., `length[n-1]`. Write an error message when it is not possible to split the string into sub-strings of those lengths. Print-out the generated sub-strings when it is possible.
For example, with:

- `str = Hello`, `n = 2`, and `length = (2, 3)`, the program may produce sub-strings (`He`, `llo`), or (`Hel`, `lo`).
- `str = Hello`, `n = 2`, and `length = (3, 4)`, there is no decomposition.
- `str = sampleTest`, `n = 3`, and `length = (2, 3, 6)`, the program may produce sub-strings (`sa`, `mp`, `le`, `Te`, `st`), or sub-strings (`sa`, `mp`, `leT`, `est`), or sub-strings (`sa`, `mpleTe`, `st`), etc.