Algorithms and Programming
14 June 2016
Part II: Program (18 point version)

No books or notes are allowed. Examination time: 100 minutes. Final program due by Friday the 17th, 12.00 p.m.; use the course portal page ("Elaborati" section) to up-load it.

Write a C application able to manipulate a catalog of several products with the following specifications.

A file stores all products, with the following format:

`productID productName productPrice #availability category`

where:

- `productID` (a string with exactly 10 characters) uniquely identifies the product.
- `productName` (a string with at most 255 characters) specifies the name of the product.
- `productPrice` (a real value) is the price of the product.
- `#availability` (an integer value) indicates the number of pieces available in the warehouses of the firm.
- `category` (a string with at most 10 characters) is the name of the category to which the product belong (e.g., food, beverages, media, etc.).

The list of categories is not made explicit, but has to be understood by reading the entire file. Moreover, even if the number of categories is unknown, it is limited to 10000.

The application has to allow for the following operations:

- Store the file (whose name is given on the command line) in a proper data structure.
- Search a product given its category and product id. The search has to look for the category first, and then, within the category, proceed to look for the right id. **The cost of the first operation has to be at most logarithmic in the number of categories, whereas the second operation has to be at most logarithmic in the number of ids within that category**. The name, price, and availability of the product have to be printed-out on standard output.
- Search the set of products sharing the same product name or the same product name prefix terminated with a `'*'` symbol. For example, a search with the name `rad*` would return all products with names `radiator` (possibly more than one, with different ids) and `radar` (possibly more than one, with different ids). The id, name, price, availability and category of all products found during the seach have to be printed-out on standard output.
- Insert a new product (with id, name, price, initial availability and category read from standard input).
- Update the number of pieces available for a specific product, given its id and the number of pieces to add to the existing ones (read from standard input).

Algorithms and Programming
14 June 2016
Part II: Program (12 point version)

No books or notes are allowed. Examination time: 100 minutes. Final program due by Friday the 17th, 12.00 p.m.; use the course portal page ("Elaborati" section) to up-load it.

1 (2.0 points)
Write function

```
void matMax (int **m, int r, int c);
```

which receives an integer matrix `m` of `r` rows and `c` columns and it prints-out on (standard output) the position (row and column indexes) of all elements larger than all (at most 8) adjacent elements. For example, give the following matrix:

	0	1	2	3
0	5	2	3	1
1	3	1	6	4
2	3	0	5	2

the function has to print-out `(0, 0)` (corresponding to value 5) and `(1, 2)` (corresponding to value 6).

2 (4.0 points)
Write function

```
int splitPhrase (char v[], char ***m);
```

which receives an array of characters `v` null terminated, and a pointer to a dynamic matrix `m` and it is able to copy the strings within `v` into different rows of `m`. The function returns the number of strings found in the array `v`, that is, the number of row of `m`. Notice that in `v` single spaces work as string separator, and that `m` is a dynamic matrix that have to be entirely allocated by function `splitPhrase`.
For example, the following lines of code:

```
char v[1000] = "This is a phrase to split into sub-strings";
char **m;
int n, i;
...
n = splitPhrase (v, &m);
for (i=0; i<n; i++)
  printf ("String number %d = %s\n, i, m[i]);
```

print-out (on standard output, and on different rows) the following set of strings: `"This"`, `"is"`, `"a"`, `"phrase"`, `"to"`, `"split"`, `"into"`, `"sub-strings"`,

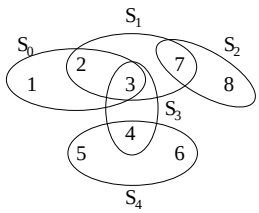
3 (6.0 points)
`S` is a set of integers included in the range $\{1,2,3,4,5,6,7,8\}$. `m` is a matrix with an undefined number of rows and 9 columns, such that it contains on each row a subset of `S` terminated by a sequence of (at least one) zero values. Write function:

```
void cover (int m[][9], int n, int k);
```

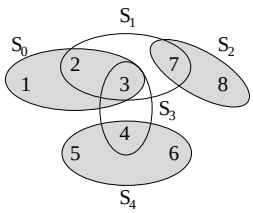
which is able to find a set of exactly `k` rows of `m`, out of `n`, such that their union is equal to `S`.
The following picture gives a graphical representation of the procedure. In this case the left-hand side matrix represents `m`, and all its subsets S_i, where `i` varies between 0 and `n-1`. The number of subsets is 5, i.e., $n = 5$, and the cover has to include exactly 3 subsets, i.e., $k = 3$. Figure (a) is the original representation of `m`, and Figure (b) represents a possible cover, i.e., $\{S_0, S_2, S_4\}$.

m:

S_0	1	2	3	0	0	0	0	0	0
S_1	2	3	7	0	0	0	0	0	0
S_2	7	8	0	0	0	0	0	0	0
S_3	3	4	0	0	0	0	0	0	0
S_4	4	5	6	0	0	0	0	0	0



(a)



(b)