Reserved cells

Ex. 1	
Ex. 2	
Ex. 3	
Ex. 4	
Ex. 5	
Ex. 6	
Tot.	

Operationg Systems

Examination task

12 February 2020

ID number _____ Surname _____ Name _____

Professor: ◯ Scanzio

It is not possible to consult texts, notes or to use calculators. The only material allowed consists in the forms distributed by the professor. Solve the exercises in the reserved spaces. Additional sheets are permitted only when strictly necessary. Report the main steps for solving exercises.
Duration: 100 **minutes.**

1. Suppose that the hard disk of a small embedded system is composed if 32 blocks of 1 MByte each, which are numbered from 0 to 31. Suppose that the operating system keeps track of the free (occupied) blocks indicating them in a vector with the value 0 (1), and that the current situation of the disk is represented by the following vector:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	1	0	0	0	0	1	0	0	1	1	1	0	1	0	0	1	0	1	0	0	1	0	0	1	1	1	0	1	0	1	1

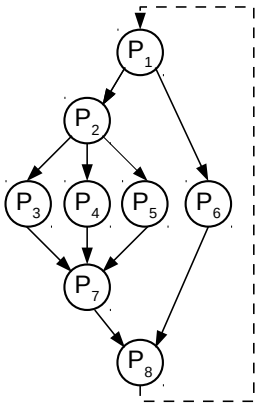
With reference to the file allocation methodologies *contiguous*, *linked*, *FAT* and *indexed*, indicate how the files `File1`, `File2` and `File3` (with dimension 4.4, 3.6 and 5.9 Mbyte, respectively) can be allocated.

Schematically report the information stored in the directory entry, and where that information is stored.

2. Two processes (P_1 and P_2) must access in mutual exclusion to a critical section called *CS*. Solve this problem with the `testAndSet` procedure, and then with the use of semaphores and the `init`, `wait` and `signal` primitives. In addition, describe the main differences (advantages and disadvantages) of the two previous solutions.

   Finally, illustrate the implementations (in pseudo-code) of the three previous functions (`testAndSet`, `wait` and `signal`).

3. A concurrent program is composed of 8 processes (P_1, P_2,...,P_8), whose temporal relationship is illustrated by the following figure:



In the case of **not cyclic** processes, report the program that implements the previous precedence graph by using only the `fork`, `wait` and `exit` system calls. In this case, do not consider the dotted arc.

In the case of **cyclic** processes (i.e., function with body `while(1)`), report the body of processes P_1, P_2,...,P_8 using the primitives `init`, `signal`, `wait` and `destroy`. Report the initialization of the semaphores, and use the minimum number of semaphores. In this case, consider the dashed arch.

4. Implement a BASH script capable to realize the "*trash*" functionality using the command line. The script manages the directory `TRASH` (which is assumed to exist and it is located in the home of the user). In this directory, the script stores all the files deleted by the user, and a hidden file `.TRASH_INDEX`, containing, for each deleted file, an entry with the following information:

   `filename absdir`

   The script must be able to perform three types of operations:

   - `--delete abspath`: deletion of the file with absolute path `abspath`. The script must check that the file to be deleted exists and is not duplicated in the `TRASH` directory, then it moves that file in `TRASH`, and it stores its name (`filename`) and the absolute path of the source directory (`absdir`) in the `.TRASH_INDEX` file.
   - `--restore filename`: restores the file with name `filename`. Use the content of the file `.TRASH_INDEX` to check for the presence of `filename`, and obtain the path of the original directory. Check the existence of this directory and move the file into it, then delete the entry related to the restored file from `.TRASH_INDEX`.
   - `--restore-all`: restores all the deleted files. This option performs the previous operation for all the files registered in `.TRASH_INDEX`, finally delete the contents of `.TRASH_INDEX`.

   Note that each invocation of the script must allow the execution of a single operation among the three described. In the case of problems during the deletion or restoration of a single file, the user must be warned with an error message, and the operation is not performed. Check for the correct number of parameters.

   Example (assuming that `trash` is the name of the script):

```
> ./trash --delete /dir1/pippo -> ("pippo" moved in TRASH, row "pippo /dir1/"
                                  added in .TRASH_INDEX)
> ./trash --delete /dir3/pippo -> ERROR  (filename "pippo" contained in .TRASH_INDEX)
> ./trash --restore pippo      -> ("pippo" moved in "/dir1/", row "pippo /dir1/"
                                  removed from .TRASH_INDEX)
```

   Suggestion: remember that the `grep` command with the `-v` option deletes all the lines that do not match the search criterion.

5. In digital image processing, the *smoothing* of an image consists in the application of a filter function whose purpose is to highlight significant patterns. Write the function:

```
void smoothing (int **mat, int r, int c);
```

which, after receiving as parameters the matrix of integers `mat` with `r` rows and `c` columns, performs the smoothing on all values, according to the following simplified algorithm.

Each value of the matrix must be replaced with the arithmetic mean of the adjacent elements (regardless of the number of adjacent elements). The function must perform the following steps:

- Define a temporary support matrix of the same size of the original source matrix (`mat`).
- Execute a number of threads equal to $(R \cdot C + 1)$.
  - The first set of $(R \cdot C)$ threads is executed immediately. Each of these threads manages a specific element of the matrix, and it is responsible for calculating the average value of the elements adjacent to it, and to store this average value in the support matrix in the corresponding position.
  - The last thread is executed only when all the previous $(R \cdot C)$ threads have terminated. It takes care of copying the temporary matrix in the original one. When this operation is finished, the `smoothing` function ends.

6. Consider the following set of processes:

Process	Arrival Time	Burst Time	Priority
P_0	0	12	2
P_1	4	18	3
P_2	8	17	1
P_3	12	13	4
P_4	16	20	5

Represent using a Gantt diagram the execution of these processes using the scheduling algorithms PS (Priority Scheduling), RR (Round Robin), and SRTF (Shortest Remaining Time First). Compute the average *waiting time* for each process and for the global set of processes. Consider a temporal *quantum* of 15 units.

Illustrate which other evaluation metrics can be used in order to compare the previously indicated scheduling algorithms.