**Problem I: Process management (25 points):**

***Part1:*** Write a C program under Unix which allows to create a process tree of **N** levels where each process of level **i** must create **2** processes (if **i** is even) and **3** processes (if **i** is odd). The initial father is in level 0, his child in level 1, and so on.  
The executable program should be called from Shell via the user command: **Tree N**.

***Part 2:*Given the following piece of code:**

**void f(int a) {**

**if(fork()) a --;**

**else a ++;**

**if(a<=0 || a%2 ==0)**

**exit(0);**

**f(a);**

**}**

1. How many processes are created following the call of the function f (1)?
2. How many processes are created in the execution of the following code:  
   **void main(){**

**f(1); f(1);}**

1. How many processes are created following the call f(4)?

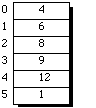
**Problem II: Memory management (25 points):**

***Part1:*** We consider a memory management system using the paging mechanism on demand. The physical memory is divided into a set of 8 frames numbered from 0 to 7. In this context, there are three processes P1, P2 and P3 whose address spaces are respectively composed of 3 pages, 5 pages and 4 pages (The pages are also numbered from 0). At a time t, the configuration of the main memory is as follows:

* For P1: pages 0 and 2 are loaded in frames 1 and 6 respectively.
* For P2: pages 0 and 2 are loaded in frames 3 and 7 respectively.
* For P3: pages 0 and 3 are loaded in frames 0 and 5 respectively.  
  The frames not mentioned above are free.

1. Represent the process page tables according to the previous configuration.
2. In this context:
   1. The process P2 requests access to the virtual address <page 4, offset 12>, describe what happens?
   2. Process P3 then requests access to the virtual address <page 3, offset 14>, describe what happens?
3. It is assumed that the pages have been loaded chronologically in the order cited i.e. We load page 0 of P1 then page 2, then we load page 0 of P2, then page 2 and similarly for P3. What will be the memory behavior for the following sequence of requests: (P2, page2), (P2, page2), (P2, page2), (P2, page2), (P1, page0), (P1, page2), (P1, page2), if the FIFO strategy is adopted? What will be the number of page faults?

***Part 2:*** Consider a memory paginated system with page size of 1 kB. The figure below shows a portion of the page table of a process (only the loaded pages are represented):



The processor generates the following virtual address (hexadecimal): **19C3**.

1. What is the size of the virtual address space?
2. What is the physical address corresponding to the virtual address generated, assuming that the memory is composed of 16 frames?

**Problem III: File System (20 points)**

***Part 1:*** Consider a file system with linked list and indexed allocation strategy as illustrated in the figure. The size of block is 8 KB.

|  |  |
| --- | --- |
|  | The disk blocks are numerated from 0 to n-1. Each opened file has an entry in the file descriptor table which is loaded in memory. This entry contains all attributes of the file and a pointer to an index block. The index block contains k pointers to other blocks. These pointers except the last points to data block. The last pointer points to another index block and so on as shown in figure. The pointer to block occupies 32 bits.  **Question:** What is the maximum size (in number of blocks) of a file in this system? Indicate the number of data blocks and index blocks |

***Part 2:*** Consider a file system on a disk that has both logical and physical block sizes of 512 bytes. Assume that the information about each file is already in memory. For each of the three allocation strategies (contiguous, linked, and indexed), answer these questions:

1. How is the logical-to-physical address mapping accomplished in this system? (For the indexed allocation, assume that a file is always less than 512 blocks long.)
2. If we are currently at logical block 10 (the last block accessed was block 10) and want to access logical block 4, how many physical blocks must be read from the disk?