Write a program in your desired object oriented programming language, that emulates a virtual memory - more importantly the mapping of the VA to PA. The program will be reading an input file which contains Virtual addresses in BIN (binary) in the order of CPU execution. For this prgogram, we will expect that the CPU will execute one program from start to finish (uniprocessing) and there will be no parallel/multi-processing.

- Assume that all the page tables of the each program are initially EMPTY.
  Note: A new page table will be created as CPU starts executing a new program.
- Apart from the binary addresses in the input file, an input file might often contain the following commands:
  - `NEW` - Start of the instructions of a new program to be executed by the CPU.

Sample Input

You can assume that the input files don't have any errors and do not deviate from the said specifications. A snippet of the sample file is as below:

```
8
4
NEW
8
101101
101110
.
.
.
NEW
16
0001100
0001101
0001110
.
.
.
```

- The pagesize, memory size and virtual memory size are in decimal - **DEC**.
- The instructions are given in binary.
- The first line in the input file indicates the number of words/instructions in each page.
- The second line in the input file indicates the maximum number of pages the main memory can hold at any given amount of time.
- Every NEW command will be followed by the size of the virtual memory for that program.
- Rest of the lines are virtual addresses to be executed in the same order as given in the input file.
- Every block of instructions seperated by the `NEW` command constitute one program. As mentioned earlier, assume that the CPU runs one program from start to finish without deviating until it encounters a "`NEW`" command. (Most likely not the case in reality).
- Each individual program will contain the number of pages right before the actual instructions.

Sample Output

The output needs to be formatted exactly as shown below. The first line of output after executing an instruction will contain the page fault rate, followed by the contents of all the page tables the CPU has encountered so far (in the order of execution). Outputs should be displayed everytime the CPU starts a `NEW` program.

```
XXX.XXX%  XX  XX  XX  XX  XX  XX  XX  XX
          XX  XX  XX  XX  XX  XX  XX  XX  XX  XX
.
.
.
XXX.XXX%  XX  XX  XX  XX  XX  XX  XX  XX
          XX  XX  XX  XX  XX  XX  XX  XX  XX  XX
          XX  XX  XX  XX
          XX  XX  XX  XX  XX  XX  XX  XX
```

Program Structure

Your program should at the very least contain a class named -

- `MainMemory` - Main memory will contain all the page tables so far.
- `CPU` - Every `NEW` program instruction being executed by the CPU will trigger a new PageTable object creation. The new page table object created will also be added to the Main memory by the CPU. The CPU class will include subroutines to convert VA to PA, to execute an instruction and also a subroutine to move a page from Virtual memory to a page frame in main memory(if needed).
- `PageTable` - The PageTable class will include functions/variables that are necessary to create and maintain a page table for the current program..

Tips

- Comment your source code appropriately, include an informative header at the top of your program, and use good coding style. Comments will be included in grading rubric as well.
- **Use meaningful names for variables and subroutines**.
- You can assume that the input file does not contain any errors in the program.
- Of course, feel free to Google things to help you. As always, do not Google solutions and make sure to *cite sources*.
- A minimum of 30% points will be deducted for programs that contain compilation errors.

Submission

Submit your source code files, either individual class files or cumulated into one python file.

Rubric

Detailed rubrics for this assignment is present on Canvas.