1. Assume that the following C code

```
int array[1000];
...
for (i = 0; i < 1000; i++)
   array[i] = 0;
```

translates into the following assembly code (the hexadecimal number in front of each instruction shows its virtual address):

```
0x1024 movl $0x0,(%edi,%eax,4)
0x1028 incl %eax
0x102c cmpl $0x03e8,%eax
0x1030 jne 0x1024
```

The first instruction moves the value zero (shown as $0x0) into the virtual memory address of the location of the array; this address is computed by taking the contents of the %edi register and adding the content of the %eax register multiplied by four to it. Thus, %edi holds the base address of the array, whereas %eax holds the array index (i); we multiply by four because the array is an array of integers, each size four bytes.

The second instruction increments the array index held in %eax. The third instruction compares the contents of that register to the hex value 0x03e8, or decimal 1000. If the comparison shows that that two values are not yet equal (which is what the `jne` instruction tests), the fourth instruction jumps back to the top of the loop.

Assume the array is located at virtual address 40000 and thus, since it's 1000 x 4 bytes long, it is located between 40000 and 44000 (not including the last byte).

Assume a virtual address space of size 64 KB and a page size of 1 KB. What is the memory reference string generated by this code?

2. Write the binary translation of the logical address 0001000010111110 under the following hypothetical memory management schemes, and explain your answer.

    a. A paging system with a 512-address page size, using a page table in which the frame number happens to be four times smaller than the page number.

    b. A segmentation system with a 2K-address maximum segment size, using a segment table in which bases happen to be regularly placed at real addresses: 22 + 4,096 x segment number.

3. A process has four frames allocated to it. (All the following numbers are decimal, and everything is numbered starting from zero.) The time of the last loading of a page into each frame, the last access to the page in each frame, and the virtual page number in each frame are as shown below (the times are in clock ticks from the process start at time 0 to the event).

Virtual Page Number	Frame	Time Loaded	Time Referenced
20	0	60	161
22	1	130	160
24	2	10	162
31	3	20	163

A page fault to virtual page 23 has occurred at time 164. Which **frame** will have its contents replaced for each of the following memory management policies?

- FIFO (first-in-first-out)

- LRU (least recently used)

- Optimal. Use the following reference string: 23, 25, 23, 25, 20, 22, 31, 23, 24.

4. Circle all that apply related to the *Before Memory was Virtual* paper:
    a. The paper describes the working set model and the improvements on it.
    b. The paper makes the case that at that time virtual memory was needed solely for better memory management.
    c. The paper describes the research path that led to a PhD thesis related to memory management.
    d. The paper makes the point that page replacement algorithms strongly influence the performance of a system with virtual memory, and recounts in great detail performance comparisons between different such algorithms.
    e. The paper makes the case that virtual memory might not be needed when enough physical memory will be available.
    f. The paper does not talk about virtual memory, but about the time before, when 640 KB of physical memory was all that was needed.

5. Suppose we have four programs:
    a. One exhibits both spatial and temporal locality;
    b. One touches each page sequentially and then repeats the scan in a loop;
    c. One references pages according to a Zipf distribution. A Zipf distribution follows the Zipf law that, applied to this context, states that a small number of pages are referenced very often while very many others are referenced rarely.
    d. One generates memory references completely at random using a uniform random number generator.

All four programs use the same total amount of virtual memory—that is, they all touch N distinct virtual pages, among a much larger number of total references.

For each program, sketch a graph showing the rate of progress (defined as number of instructions executed per unit of time) of each program as a function of the physical memory available to the program, from 0 to N. Assume the page replacement algorithm approximates least recently used.