The starting point is problem 41 of Tanenbaum, page 267. For the other algorithm, choose any other, not necessarily the most difficult.

However, instead of reading the number of the referenced page from a file, the program should successively process two hexadecimal numbers from standard input. The first number is 0, 1, or 2 to signify whether the reference is a read, write or instruction fetch respectively. The second number is the virtual address. Hint: To read hex in C++:

```
#include <iostream>
...
  cin >> hex >> refType >> hex >> vaddr;
...
```

For simplicity, and in deference to Intel, fix the page size to be 4K. Define the number of page frame parameter with a global const int so that you can easily change it and recompile your program to simulate different size memories. An even better way, harder to design though, is to make all the data structures dynamic so that the parameter(s) can be set at run time.

In addition to writing to stdout the listing described in Tanenbaum, the program should accumulate all relevant statistics from the simulation: number of page frames in the simulated memory, number of references processed, number of simulated page faults, number of dirty pages written, number of pages evicted, etc. Each statistic should be printed on in decimal on line with an explanation; eg. “Number of page frames=1000”.

The program should test for command line argument “-q” and if it is given, omit the reference-by-reference listing.

The program should (ideally) accomodate a million or so references. Test it first with just a few, and reproduce the examples in Tanenbaum.

The trace files are available on acunix as .din.Z files under the pathname “acs400/public/Traces. Do not keep copies of them in your acunix account because they will use up your quota. To supply them to your tracer program for testing, use the pipeline: (What does zcat do?)

```
zcat "acs400/public/Tracer/cc1.din.Z | tracer -q
```

Your program should handle each full trace in less than 10 minutes (That’s my guess but it might be modified as people begin to try this out on acunix.)

You will probably need a more sophisticated data structure to obtain reasonable performance on the full traces. One idea is to simulate a TLB too.

Another is to use a hash table (in addition to the linked list or other data structure for simulating the algorithm) to quickly locate the record for each page or determine it doesn’t exist.

Investigate how each algorithm behaves on some of the supplied traces (or shorter versions of them) plus how it behaves on a random sequence of addresses. (Use a random number generator subroutine.)

For the grading, write a report that explains to the TA and I how we should compile and test your program, and reports what you did and what you found out. The last 25% of the credit for each of the 3 algorithms will be granted if your programs handle the full million addresses.

Turn in your work to the project named “tracing”