# CS 414 Summer 2000
# Problem Set 2
### (due) Tuesday, August 8th (in class)

1. Consider a virtual memory system, which has a page frame of size *p* bytes. Let *e* bytes be used to store each entry in the page table.
   (a) Assuming that the average size of a process is *v* bytes, give an expression for the average per process space overhead (wasted memory space) to use this virtual memory system. Also give an expression for the optimal page size considering the space overhead.
   (b) For a system using 4 bytes to store a page table entry, having an average process size of 1MB, compute the value of optimal page size considering the space overhead.
   (c) Assume that each disk operation involves a seek time of *s* sec and an average rotational latency of *r* sec. Let the disk transfer time be *t* bytes/sec. Give an expression for the time overhead for swapping one page. What would be the optimal page size to minimize this paging time? Does having this page size really save time? Why?
2. There is an interesting way to model the page replacement algorithms using a concept called **subset property**. Let *s* denote the ordered set of page requests generated by a process, i.e., *s* represents a reference string. Let us denote by $M_n(s, i)$, the set of pages in memory after the $i^{th}$ page request has been handled. Here *n* denotes the number of page frames in memory. A page replacement algorithm is said to satisfy the subset property if and only if for every reference string *s*, $\boldsymbol{M_n(s, i) \subseteq M_{n+1}(s, i)}$, after handling every request *i* in the reference string *s*.
   (a) For the reference string s = 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5, show that the LRU algorithm satisfies the subset property for n=3. (n+1 = 4).
   (b) Show that any algorithm that satisfies the subset property will not exhibit Belady's anamoly.
   (c) Show that the LRU algorithm always satisfies the subset property.
3. Cornell University is designing a database system to maintain the records of its students. Each student will have a unique id number called cornellid. Most of the accesses to this database involve searching for the record of a particular student using his cornellid.
   (a) What kind of a file system implementation would you suggest for Cornell University. Why? Assume that all records are of fixed size and that the record size is equal to a block size.
   (b) The students are also associated with a unique netid. Sometimes, the student records need to be searched using the netid instead of the cornellid. What modifications to your file system would you suggest to accommodate this efficiently?
   (c) Sometimes, the student records need to be fetched using the last names of the students. Can you efficiently incorporate this also into your system?