Part 1, Theory

Problem 1, Optimization

a) Branches can be a preformance problem due to pipelining in the processor. This means that every time a branch happens the processor might have to flush when it preforms a branch - which takes time. In order to solve this the processor tries to predict whether the branche will happen or not, and start working on the instructions it think will happen next.

Branching will also cause some delay due to instruction fetching and cache misses.

b) Spatial locality is when you use data that lies relatively close to each other. Temporal locality refers to when you reuse data within a small time interval.

c) Spatial locality: When a data element is loaded from memory to cache, data that lies close to this element is also loaded into the cache - since it's a high chance it will be used within a small time interval.

Temporal locality: When the cache has to flush data back to memory because it requires the space for other data elements, it usually flushes the data that has been least recentley used. Since it's a high chance that you'll reuse data that you've just used.

Problem 2, Caching

a)

gcc version: 3.4.3

Struct_of_arrays:

- Real: 2.844 sec
- User: 2.747 sec
- Sys 0.031 sec

Array_of_structs:

- Real 2.638 sec
- User: 2.542 sec
- Sys: 0.015 sec

b) Array_of_struct is the best layout for performance. The reason for this is that during the for-loop is accesses all the elements of the same struct at the same time. Since all the elements are in the same struct they're stored seqencally, and probably cached together. Struct_of_arrays make one array for each of the letters, and thus accesses several different arrays at the same time during each iteration of the for loop. These arrays are not stored sequencally, are will take longer time to fetch from memory.

Problem 3, Basic Valgrind

Issue: Allocate more memory than what is needed.
Bugs/errors:

- Loses reference to allocated memory, and does not free it.
- Allocates memory, but does not use it. Instead set the char-string to a constant.

Problem 4, Intermediate Valgrind

- Does not free the allocated memory. (error)
- Allocated more memory than needed. (issue)
- Could use mem[i+64] = i, instead of accessing the array again on line 11. (Would give the same result, without extra memory access). (issue)
- Everything could be run in one for loop, reducing the workload. (issue)
- Inserts integers into a char array. Would be a problem for larger values.

Problem 5, Intermediate Valgrind

- The variable lastChar might be unasigned, which might cause problems.
- If the parameter string is larger than 10, then strcpy will do an invalid write since it's copying outside of mem's range.
- The first foor loop will read 1 byte outside of mem, due to it starting on index 10 (should be 9).
-