6 marks **Question 1.** Short Answer

2 marks **Part (a)** Explain how a bottom-up traversal of a hierarchical page table saves time as compared to a top-down traversal.

1 mark **Part (b)** In what situation does a bottom-up traversal of a hierarchical page table need to "fail over" to a top-down traversal?

CONTINUED ON PAGE 3

2 marks **Part (c)** A system has 64-bit virtual and physical addresses (8-byte pointers) and pages of size 64k (2^{16}) bytes. You've done some benchmarking and discovered that most processes use, on average, 3 pages in the stack, 21 pages in the text segment (code and globals), and 40 pages in the heap segment. State how you would implement the page table and justify your implementation using the benchmarking data.

1 mark **Part (d)** For the 64-bit system described in the previous subquestion, how many memory accesses are required to perform a load (or store) in the vast majority of cases? In the worst case, how many memory accesses are required?

CONTINUED ON PAGE 4

*[Use the space below for rough work. This page will **not** be marked, unless you clearly indicate the part of your work that you want us to mark.]*

10 marks **Question 4.** Virtual Memory

Consider the following two versions of a program that finds the longest common subsequence between two strings. The difference between the two is whether the matrix is updated by row or by column.

Version 1

```
#define MAX_LEN 256
S1 = ".....";     # S1 is MAX_LEN - 1 characters.
S2 = ".....";     # S2 is MAX_LEN - 1 characters.
int len[MAX_LEN][MAX_LEN]; # Assume this matrix is zerofilled.

int lcs() {
  int S1pos, S2pos;
  for(S1pos = 1; S1pos < MAX_LEN; S1pos++) {
    for(S2pos = 1; S2pos < MAX_LEN; S2pos++) {
      if (S1[S1pos - 1] == S2[S2pos - 1]) {
        len[S1pos][S2pos] = len[S1pos - 1][S2pos - 1] + 1;
      } else {
        len[S1pos][S2pos] = max(len[S1pos - 1][S2pos],
                                len[S1pos][S2pos - 1]);
      }
    }
  }
  return len[MAX_LEN - 1][MAX_LEN - 1];
}
```

Version 2

```
#define MAX_LEN 256
S1 = ".....";     # S1 is MAX_LEN - 1 characters.
S2 = ".....";     # S2 is MAX_LEN - 1 characters.
int len[MAX_LEN][MAX_LEN]; # Assume this matrix is zerofilled.

int lcs() {
  int S1pos, S2pos;
  for(S2pos = 1; S2pos < MAX_LEN; S2pos++) {
    for(S1pos = 1; S1pos < MAX_LEN; S1pos++) {
      if (S1[S1pos - 1] == S2[S2pos - 1]) {
        len[S1pos][S2pos] = len[S1pos - 1][S2pos - 1] + 1;
      } else {
        len[S1pos][S2pos] = max(len[S1pos - 1][S2pos],
                                len[S1pos][S2pos - 1]);
      }
    }
  }
  return len[MAX_LEN - 1][MAX_LEN - 1];
}
```

CONTINUED ON PAGE 13

The two versions of the program are being run on a system where the page size is 1024 bytes (1 Kbyte) and an int is 4 bytes. Assume the code is stored in virtual page 0, and that each string and array is stored starting at the beginning of a virtual page. Arrays are stored by row. (i.e., the elements of row 0 are stored contiguously, followed by the elements of row 1, etc.) When indexing into the array, the row is the first index. The second index refers to the column.

In all cases, assume that the program in the question is the only program running on the machine.

1 mark **Part (a)** How many pages are required to store array `len` in either version of the program?

2 marks **Part (b)** Suppose there are 1024 physical page frames and a FIFO page replacement algorithm is used. How many page faults are incurred by Version 1 of the program? (If the answer is large, an approximation is acceptable.) Briefly explain your answer.

2 marks **Part (c)** Suppose the machine has 16 physical frames and a LRU page replacement algorithm is used. How many page faults are incurred by Version 1 of the program? (If the answer is large, an approximation is acceptable.) Briefly explain your answer.

CONTINUED ON PAGE 14

2 marks **Part (d)** Suppose the machine has 16 physical frames and a FIFO page replacement algorithm is used. How many page faults are incurred by Version 1 of the program? (If the answer is large, an approximation is acceptable.) Briefly explain your answer.

2 marks **Part (e)** Suppose the machine has 16 physical page frames and a LRU page replacement algorithm is used. How many page faults are incurred by Version 2 of the program? (If the answer is large, an approximation is acceptable.) Briefly explain your answer.

1 mark **Part (f)** Which version of the program is likely to run faster? Why?

CONTINUED ON PAGE 15