COS126: General Computer Science	Spring 1996

Third Midterm Exam: Solutions

Date: Monday, May 6, 1996

Notes: *If you wish to question your grade on any question, write your inquiry on a piece of paper and hand it to your preceptor.*

We would like to make a general comment on our grading scheme on Questions 1 and 2. At this point in the course we heavily penalize any program that displays ignorance of some basic feature of C, such such as how to write loops or make function calls, or the fact that a `return` *statement terminates the execution of the function. We are more tolerant of errors that were clearly made in the "heat of the exam."*

Problem 1, 25 points: A Correct Solution:

```
int counteven(struct tree *t)
{
 int i=0;

 if (t == NULL)
    return 0;
 i = counteven(t->left) + counteven(t->right);
 if ((t->value % 2)==0)
    return i+ 1;
 else
    return i;
}
```

The following is an explanation of the grading.

1. Anybody who used recursion in a basically correct way got 15 points. Minor syntax errors (missing variable declarations, parentheses etc.) cost 2 points each. Many students made the assumption that if the function does not explicitly return a value, then the default value is 0. This is not true for all C compilers, so 2 points were deducted.
2. Using a global variable in this question is not incorrect but extremely bad programming practice. The penalty is 5 points.
3. Failing to check that a pointer is not `NULL` before dereferencing (i.e., using it in an expression like `t->value`) will lead to core-dumps, and costs you 5 points.
4. Many students wrote something along the following lines.

   ```
   int counteven(struct tree *t)
   {
    int i=0;
   ```

```
 if (t == NULL)
    return 0;
 counteven(t->left);
 counteven(t->right);
 if ((t->value % 2)==0)
    { i++; return 1;}
 else
    return 0;
}
```

This is incorrect and shows a very basic ignorance of scope rules and return values. We still gave this 10 points, but similar efforts on the final exam will earn a 0.

5. Anybody who got 5 or less on this question exhibited even more basic ignorance of basic rules of programming (see above). For example, many students had variations on the following piece of code

```
 if ((t->value % 2)==0)
   return 1;
 else
   return 0;

 t= t->left;
.....
```

Note that the line `t= t->left;` and any line after that will never be executed. Failure to recognize this shows ignorance of the meaning of the `return` statement.

Problem 2: 25 points

```
#include <stdio.h>
#include <ctype.h>
void main()
{
 FILE *fp;
 int c, state=0, count=0;
 fp =fopen("myfile.c", "r");
 if(fp == NULL)
  { printf("Myfile not found\n"); return ;}
 while((c= getc(fp))!= EOF)
  {
   if ((c > 'A')&&(c<'Z'))
      c= tolower(c);
   /*tolower(x) changes an uppercase x to lowercase */
   /* convention: state ==1 means last letter seen was 'n'*/
   /* state==2 means last two letters seen were 'n' and 'o' */
   switch (c)
   {
```

```
      case 'n':
          state=1;
          break;
      case 'o':
          if (state==1)
           state=2;
          else
            state=0;
          break;
      case 't':
          if (state == 2)
            count++;
          state = 0;
          break;
      default:
          state=0;
      }
     }
  printf("Not was found %d times in myfile\n", count);
}
```

1. When your program tries to open a file for reading, it should check that the pointer returned is not `NULL`. Failure to do so can result in core-dumps. We penalized core-dumps with -5 points. Another common reason for core-dumps: overstepping an array bound.
2. Any program that correctly read the file (and didn't show any major lack of understanding of C) got at least 7 points.
3. Many students tried to read the entire file into an array. This meant that they had to impose an arbitrary upperbound on the file size. We consider such a solution quite unacceptable, and gave it no more than 11 points.

   We remind you: *array sizes have to be declared at compile time.* Many students tried to define an array as follows:
   `char myarray[];`
   The hope clearly was that this would define `myarray` to be an array of flexible size, which would "grow" automatically at run-time. As a matter of fact, this declaration declares `myarray` to be a pointer; and thus referring to `myarray[2]` in the program will cause a core-dump. (We deducted some points for this confusion, and will deduct even more points on the final for similar errors).

   Some students defined `myarray` to be a pointer but tried to use string primitives such as `strcat` to achieve the above goal. This also does not work.
4. Any reasonable program that didn't try to read the file into an array got at least 15 points.
5. Many programs skipped characters in the file. In other words, they would detect some "not"s in the file but not all. The most frequent cause of this error was sloppy loop control; for example, by including a `getc()` statement as part of

the header of the `for` loop, but then ignoring the result of this `getc()`. We are tolerant of such errors since they are clearly caused by the time-pressure of the exam, and deducted only 3 points for them (unless they revealed some bigger problem in your understanding).

6. Some programs used `goto`s, which we explicitly forbade you from using in this class. We deducted only 4 points, but beware in future.

Problem 3: 25 points Part (a): 15 points

```
Memory Access:  0A11ABCD  --> PAGE FAULT
                page no   in memory   physical location
                ---------------------------------------
                  0A11       yes            0A7
                  0A16        no            disk

Memory Access:  0A12ABCD  --> NO PAGE FAULT

Memory Access:  0A110123  --> NO PAGE FAULT

Memory Access:  0A160000  --> PAGE FAULT
                page no   in memory   physical location
                ---------------------------------------
                  0A14        no            disk
                  0A16       yes            FFF

Memory Access:  0A14AA11  --> PAGE FAULT
                page no   in memory   physical location
                ---------------------------------------
                  0A14       yes            FFF
                  0A16        no            disk
```

Part (b): 10 points

If physical memory is quadrupled, the total size becomes: $2^{28} * 2^{2} = 2^{30}$. So we now need 30 bits to specify a physical address. The 32-bit virtual address is more than adequate since it is longer than the 30-bit physical address. Recall that one of the purposes of virtual memory is to give the illusion that memory is unlimited – because of this, the size of the virtual address must be at least as large as the size of the physical address.

We now need 14 bits to specify physical page numbers with this larger memory, since the total number of physical pages is now: 2^{30} / 2^{16} = 2^{14}.

1. Each answer in part (a) was worth 3 points. Points were deducted if page faults were not correctly identified, or the page table was incorrectly modified. Some students forgot that the page table changes were cumulative – i.e. if a particular memory access results in a page fault, then the changes to the page table are visible to subsequent accesses. Points were deducted for this mistake.

2. The first part of (b) (determining whether or not the 32-bit virtual address is adequate) was worth 5 points. Basically, you needed to specify that 32-bits is adequate because it can reference at least as much memory as there is in physical memory. You got all 5 points if you said this. If you said that the 32-bits is adequate without backing it up properly, you got 4 points. If you said that 32-bits is inadequate, you only got 2 points.

   Some students misinterpreted the definition of quadruple. If physical memory is quadrupled, then its size is multiplied by 4 – instead, some students multiplied the exponent by 4, resulting in a total memory size of 2^{112}. I gave partial credit if you said that 32-bits is inadequate because it references less memory than 112 bits does.

3. The second part of (b) (determining how many bits are needed to specify a physical page number) was also worth 5 points. You got 2 points if you gave the incorrect answer, but showed some work. You got 1 point if you gave the wrong answer and didn't show any work. Again, I gave partial credit if your incorrect answer was based on your misinterpretation of quadruple.