

Midterm Exam 2 Solutions

C Programming

Dr. Beeson, Spring 2009

April 16, 2009

Instructions: Please write your answers on the printed exam. Do not turn in any extra pages. No interactive electronic devices of any kind are allowed during the exam (no computer, no calculator). You can use your music player (if nobody else can hear it) if you put it on shuffle and don't punch any of its buttons during the exam. The exam is open notes, open book—you can use any printed materials that you brought with you. *Scoring:* Five points per problem, total 100.

1. Recall the function `void add(char *x, char *y, char *ans)` that you used in two of your homework assignments. What is the correct way to call this function to have it carry out the addition $45 + 78 = 143$?

(d) `char ans[40]; add("54", "87", ans);` *We need to use an array `ans` so there will be space, and we need to reverse the order of digits because that is how `add` expects them.*

2. Suppose you have to write `void BitReverse(int *x)` that reverses the bits of `*x`. Suppose, in the code, you have declared an integer variable `t` and written code that makes `t` the reverse of `*x`. Now, what will be the final line of code?

(c) `*x = t;` *Write the answer where the calling function expects it.*

I saw that a lot of students answered with `strcpy`. But `strcpy` needs two pointers to char—how did you think it would take an `int`? Some other students answered `x = &t`, but both `t` and `x` are going out of scope as soon as this function exits, so changing the value of either one is not going to have any effect, let alone the desired effect.

3. It is required to write a program that takes two numbers on the command line and prints out their product. Example session:

```
C:> multiply 45 78
35410
```

The program terminates after working one multiplication problem. Fill in the blanks below to write an eight-line program to accomplish this. Make the code match the given comments. Longer answers and answers that do not follow the comments will not be read.

```
int main(int argc, char *argv[]) // define main
{ int m,n; // declare two integer variables
  if(argc != 3) // check if two arguments were supplied on the command line
    { printf("Usage: %s n m", argv[0]); // if not print a "usage" message
      return 1; // and exit
    }
  sscanf(argv[1], "%d", &m); // convert the first command-line argument to a number
```

```

    sscanf(argv[1], "%d", &n);           // convert the second command-line argument to a number
    printf("%d", n*m);                  // print the product
    return 1;                            // main has to return something
}

```

4. Give `printf` commands to print out

(a) an int `n`

```
printf("%d", n);
```

(b) given strings `FirstName` and `LastName`, print them out formatted as `LastName, FirstName`.

```
printf("%s, %s", LastName, FirstName);
```

(c) a double `x`, with two digits showing after the decimal place, e.g. 4.00 or 4.02 or 246.19.

```
printf("%.02lf", x);
```

(d) a double `x`, using exponential notation as 4.0e-12.

```
printf("%le", x);
```

(e) an unsigned long `x`.

```
printf("%lu", x);
```

(f) an unsigned integer `x`.

```
printf("%u", x);
```

(g) a long long `x`.

```
printf("%lld", x);
```

5. Suppose we have arrays `char *countries` and `double values`, and we want to print a table like the following one, with the countries starting in the same column on each line, and the decimal points lined up in the second column

Australia	4.235
Japan	8.410
United States	18.940

Give a `printf` command that will print the k -th line of this table.

```
printf(" %-18s%10.3lf\n", countries[k], values[k]);
```

You need the hyphen at the first to get the first column left-justified; the field width of 10 makes the right-justification default work for the floating point values; three digits after the decimal point. Incidentally, I could not manage to line up the decimal points AND not print the trailing zeroes in one printf command.

6. Give one line of code to convert a string x to an int n (assuming it does convert, and that n and x have already been declared).

```
sscanf(x, "%d", &n);
```

7. Give two lines of code to convert an int n to a string x (assuming that n has been declared, but the first line of your two lines should declare x appropriately).

```
char x[40];
sprintf(x,"%d",n);
```

8. Declare an array x of strings, and initialize it to contain the three strings "cat", "dog", "giraffe". Use only one line of code to do this.

```
char *x[] = {"cat", "dog", "giraffe"};
```

9. (a) Declare a 10 by 10 array of doubles.

```
double x[10][10];
```

(b) How many bytes will the C compiler use for this array?

800 (because this is 10 times 10 times sizeof(double))

10. Fill in the two blank lines in the following code with recursive calls, so that the function `find` fulfills the specification given in the comments.

```
find(int x, int data[], int p, int q)
// find and return some index k such that x = data[k] and p <= k <= q,
// or return -1 if there is no such index.
// assumes data[k] is defined for p <= k <= q and data is sorted in ascending order
{ int m = (p + q)/2;
  if(x == data[m])
    return m;
  if(p==q) return -1;
  if(x < data[m]) // then search the left half of the array
    return find(x,data,p,m-1);
  else
    return find(x,data,m+1,q) // then search the right half of the array
}
```

11. define a macro `SQRT(x)` so that we just type `SQRT(x)` instead of the more complicated `(int) sqrt((double) x + 0.1)` when we want to take the square root of an integer.

```
#define SQRT(x) ((int) sqrt((double) (x)+ 0.1))
```

12. Some program uses several arrays that should all have the same dimension N . Give one line of code that will set the value of N in just one place, rather than in several different array declarations.

```
#define N 100
```

13. What is the difference in effect between the following two lines?

```
#include <stdio.h>
#include "stdio.h"
```

(b) The first one looks for `stdio.h` first in the system include file directory, then while the second one looks in the program's directory. (Actually, the wording of choice (b) on the exam is not literally correct—but it was the closest to correct. This wording is accurate.)

14. Suppose I want to compile three versions of my (large, multi-file) program, one for the Mac, one for Linux, and one for Windows. Most of the source code will be the same but some parts will be different. I want to be able to recompile for a different operating system after changing just *one line of code*.

(a) Where should that one line of code be?

In a header file include in all the files where there are different versions of the code

(b) What should it be, when I want to compile for the Mac?

```
#define MAC
```

(c) What preprocessor commands will be used to surround the parts of the code that are used only when compiling the Mac version?

```
#ifdef MAC
#endif
```

15. Which of the following would be appropriate in a header file `myfile.h` (more than one answer is correct, so circle all correct answers).

(b) prototypes of functions defined in `myfile.c` intended for use in other files.

(e) `# define` commands for identifiers needed in several different files.

16. Which of the following divides x by 2 (assuming x is an unsigned int or unsigned long)?

(b) `x = x >> 1;`

17. Which of the following tests if x is odd (assuming x is a signed or unsigned int or long)

(a)

```
if(x & 1)
```

18. How do I declare a function f that takes an `int` and returns an `int` if I want to be sure that f is accessible only in the file where it is declared?

```
static int f(int n)
```

19. Give an expression that can be put inside `if` to test whether a long long x will fit into an `int` or not. Write this expression so that it makes no assumptions about how many bytes `ints` and `long longs` require, (so it will still be valid twenty years from now).

The easiest solution is `x == (int) x`.

```
if( x & (1LL << 8*sizeof(int)) )
    { // then x fits in an int
```

If you forget to write 1LL, and write just 1 instead, then the left-shifted expression will always be zero and the test will always fail. If you write 1L instead of 1LL (as I did when I first posted these solutions!) then the left-shift is done using long arithmetic, not long long arithmetic, and hence has undefined behavior; I think on most systems it is done mod 32, so 1L j 32 comes out to be 1.

For an unsigned long long, you could right-shift by 8 sizeof(int), but that won't work for a long long, as sign bits may be shifted in. Behavior of right-shift on signed objects is not specified.*

20. What will be printed by the following code the third time *f* is called?

```
int f(int k)
{ static int count = 0;
  ++count;
  printf("%d", count)
}
```

Answer: 3 will be printed.