Spring 2014 Introduction to Computer Programming (COP 3223) Test #3 Section 4

SOLUTION

1) (5 pts) The geometric mean of two numbers, a and b is $\sqrt{ab}$. Complete the function below so that it returns the geometric mean of its two parameters, *a* and *b*. You may assume that the math library is included and that both input parameters are positive. The function that calculates a square root is sqrt and the function that calculates an exponent is the pow function. In particular pow(x, y) returns x^y.

```
#include <math.h>

double geomean2(double a, double b) {

     return sqrt(a*b); // 1 pt return, 1 pt sqrt, 3 pts a*b
}
```

2) (8 pts) The geometric mean of three numbers, a, b and c is $\sqrt[3]{abc}$. Complete the function below so that it returns the geometric mean of its three parameters, *a*, *b*, and *c*. You may assume that the math library is included and that all input parameters are positive.

```
#include <math.h>

double geomean3(double a, double b, double c) {

     return pow(a*b*c, 1.0/3); // 1 pt ret, 1 pt pow, 3 pts a*b*c
                               // 3 pts 1.0/3
}
```

3) (10 pts) In general the geometric mean of *n* numbers, x_1, x_2, …, x_n is $\sqrt[n]{x_1x_2\ldots x_n}$. Complete the function below so that it returns the geometric mean of the first *n* values in the array, *values*, where *values* and *n* are the formal parameters, respectively. Once again, you may assume that the math library is included and that n and all the values in the input array are positive. Do not worry about any overflow or imprecision issues.

```
#include <math.h>

double geomean(double values[], int n) {

    double prod = 1;                 // 1 pt
    int i;
    for (i=0; i<n; i++)              // 2 pts
        prod *= values[i];           // 3 pts
    return pow(prod, 1.0/n);         // 4 pts


}
```

4) (10 pts) Complete the function below so that it returns 1 if and only if the first *n* values of the input array *values* is in ***ascending*** sorted order. (Note: In order for an array to be in ascending order, each subsequent value must be ***greater than*** the previous value. Thus, an array storing 2, 3, 6, 8 and 12, in that order, is in ascending sorted order, but an array storing 3, 6, 8, 8 and 9 is NOT in ascending sorted order, since the two 8s are in a row.)

```
int isSorted(int values[], int n) {

    int i;                              // 1 pt
    for (i=0; i<n-1; i++)               // 3 pts
        if (values[i] >= values[i+1])   // 3 pts
            return 0;                   // 1 pt
    return 1;                           // 2 pts for waiting till here

}
```

5) (25 pts) A border square in a two-dimensional array is one that is in the top row, bottom row, leftmost column or rightmost column. Complete the function below so that it returns the number of border squares in the input array that store the value 0. The function takes in the two dimensional array. You may assume that its dimensions are NUMROWS x NUMCOLS, where these two values are the constants declared below.

```
#define NUMROWS 20
#define NUMCOLS 25

int numZeroBorderSqs(int grid[][NUMCOLS]) {

    int cnt = 0, i;
    for (i=0; i<NUMCOLS; i++) {
        if (grid[0][i] == 0)
            cnt++;
        if (grid[NUMROWS-1][i] == 0 && NUMROWS > 1)
            cnt++;
    }

    for (i=1; i<NUMROWS-1; i++) {
        if (grid[i][0] == 0)
            cnt++;
        if (grid[i][NUMCOLS-1] == 0 && NUMCOLS > 1)
            cnt++;
    }

    return cnt;
}

// Grading: 5 pts for doing row 0
//          5 pts for doing last row
//          5 pts for doing col 0
//          5 pts for doing last col
//          5 pts for not overcounting corners.

// 15 pts for counting all, 8 pts for double loop
```

6) (12 pts) What is the output of the following program?

```
#include <stdio.h>

int f(int a, int b) ;

int main() {
    int a = 5, b = 12, c = 7;
    b = f(a+b, a+c);
    printf("a=%d, b=%d, c=%d\n", a, b, c);
    c = f(c, b);
    printf("a=%d, b=%d, c=%d\n", a, b, c);
    return 0;
}

int f(int a, int b) {
    int c = 3*a - b;
    b = 24 - a;
    a = c%a;
    printf("a=%d, b=%d, c=%d\n", a, b, c);
    return 2*b + a;
}
```

```
a=5, b=7, c=39
a=5, b=19, c=7
a=2, b=17, c=2
a=5, b=19, c=36 // Grading - 1 pt each grade exactly.
```

7) (10 pts) What is the output of the following program?

```
#include <stdio.h>

int main() {

    int i, array[10];
    for (i=0; i<10; i++)
        array[i] = (3*i+2)%10;
    for (i=0; i<5; i++) {
        int temp = array[i];
        array[i] = array[9-i];
        array[9-i] = temp;
    }
    for (i=0; i<10; i++)
        printf("%d ", array[i]);
    printf("\n");

    return 0;
}
```

```
9 6 3 0 7 4 1 8 5 2 // Grading - 1 pt each grade exactly.
```

8) (12 pts) What is the output of the following program?

```
#include <stdio.h>

int f(int* a, int b) ;

int main() {

    int a = 5, b = 12, c = 7;
    b = f(&b, a+c);
    printf("a=%d, b=%d, c=%d\n", a, b, c);
    c = f(&c, b);
    printf("a=%d, b=%d, c=%d\n", a, b, c);
    return 0;
}

int f(int* a, int b) {
    int c = 3*(*a) - b;
    b = 24 - (*a);
    *a = c%(*a);
    printf("a=%d, b=%d, c=%d\n", *a, b, c);
    return 2*b + (*a);
}
```

```
a=0, b=12, c=24
a=5, b=24, c=7
a=-3, b=17, c=-3
a=5, b=24, c=31  // Grading - 1 pt each grade exactly.
```

9) (5 pts) Give one example of data that is best stored in a two-dimensional array, in your opinion. Do not use tic-tac-toe, connect 4 or minesweeper as these were examples already given in class. Explain why a two dimensional array is an appropriate form of storage for your example.

Imagine storing a computer model of the depth of a lake. You could break the lake up into squares of fixed sizes and for each square, in a two dimensional array, store the average depth of the lake in that square. This model would be useful to calculate a number of different things, such as the total volume of water in the lake. Here is a small example of what you might see:

0	**0**	**1**	**1**	**2**	**3**	**3**	**2**	**1**	**0**
0	**1**	**1**	**2**	**4**	**8**	**6**	**4**	**2**	**1**
1	**1**	**2**	**3**	**7**	**9**	**5**	**4**	**1**	**0**
1	**0**	**1**	**2**	**6**	**4**	**2**	**3**	**2**	**0**
0	**0**	**0**	**1**	**3**	**1**	**1**	**1**	**1**	**0**

This example is typical, representing shallow borders and a deeper center.

10) (3 pts) UCF's Programming Team just finished in second place in the North American Invitational Programming Contest (NAIPC). One of the teams they beat was University of Lethbridge. In what continent is University of Lethbridge located?

North America