Fall 2011 COP 3223 Section 3 Final Exam Solution
Short Answer Section Version A

1) (3 pts) Write a single line of C code that prints out the following to the screen:

```
\\\\\\
" Hi "
\\\\\\
```

printf("\\\\\\\\\\\\\\n\" Hi \"\n\\\\\\\\\\\\\\");

Grading: 1 pt printf, 1 pt \\, 1 pt \n, \"

2) (2 pts) What are the values of the following arithmetic expressions?

a) `4 + 5*8` **44**

b) `27/(7 + 3)` **2**

c) `16 – 2*(10 – 4*(80000%8 + 2 – (18/11)))` **4**

d) `37%(3 + 4 – 3/4 + 2*4 + 5)` **17**

// Grading: ½ a point for each one, round down.

3) (2 pts) What's wrong with the segment of code shown below? (Assume all variables used in it are previously declared and given appropriate values.) What is a quick fix for the problem?

```
while (choice != 1 || choice != 2) {
    printf("That is not valid, please enter again(1,2).\n");
    scanf("%d", &choice);
}
```

There is an infinite loop. This can be fixed by changing || to &&. (Grading: 2 pts for infinite loop, 1 pt for fix.)

4) (2 pts) What are the values of the following expressions that include calls to math library functions? (Answers will be considered correct if they are within .01 of the actual value.)

a) `pow(.25, -3)` **64**

b) `sqrt(pow(6,2) + pow(4,3))` **10**

Grading: 1 pt each

5) (2 pts) In the code segment below, fill in the expression so that num gets set to a random integer in between 200 and 499, inclusive.

```
srand(time(0));

int num = 200 + rand()%300 ;
// Grading: 1 pt for rand()%100, 1 pt for + 100.
//          1 pt for rand()%500
```

6) (6 pts) A range of consecutive integers is denoted by a starting and ending value, where the starting value is less than or equal to the ending value. For example, the range [18, 22] includes the integers 18, 19, 20, 21 and 22. Consider the task of writing a function that is given two ranges that determines whether or not the two ranges intersect. For example, [18, 22] and [13, 19] do intersect, but [33, 88] and [90, 2000] do not. Complete the function below so that it returns 1 if the two ranges specified intersect, and 0 if they do not. The first range will be [start1, end1] and the second range will be [start2, end2], where start1, end1, start2 and end2 are the four formal parameters to the function. Make sure to fill in the code according to the comments given.

```
int has_intersection(int start1,int end1,int start2,int end2) {

    // [start1, end1] begins first.
    if (start1 < start2 ) {  // 2 pts

        if (start2 <= end1)  // 2 pts, 1 pt off if < instead
            return 1;        // total for both…
        else
            return 0;
    }

    // [start2, end2] begins first.
    else {

        if (start1 <= end2 ) {     // 2 pts
            return 1;
        else
            return 0;

    }
}
```

7) (2 pts) What does sum equal at the end of the following code segment?

```
int sum = 0, i;
for (i=15; i<25; i+=3)
    sum += i;
```

78 (2 pts for correct answer, 1 pt for 54, 0 otherwise)

8) (2 pts) Write a single line of C code to open the file “statistics.txt” in write mode. Name your file pointer ofp.

```
FILE* ofp = fopen("statistics.txt", "w");
// Grading – 1 pt for FILE* ofp =, 1 pt for rest
```

9) (4 pts) What is the output of the following program below?

```
#include <stdio.h>

int main() {

    int a[] = {6, 3, 5, 1};
    int b[] = {8, 2, 7, 4};

    int c[8], i = 0, j = 0, k = 0;
    while (i < 4 || j < 4) {
        if (a[i]%2 == 1 || j == 4) {
            c[k] = a[i];
            i++;
        }
        else {
            c[k] = b[j];
            j++;
        }
        k++;
    }

    for (i=0; i<8; i++)
        printf("%d ", c[i]);

    return 0;
}
```

8 2 7 4 6 3 5 1 (Grading: ½ pt per number, round down)

10) (4 pts) Write a segment of code that declares and initializes a 4 x 4 integer array so that the values in each of the four rows is [1,2,3,4], [5,6,7,8], [9,10,11,12], [13,14,15,0], using nested for loops.

```
int board[8][8];                        // 1 pt
int i, j;
for (i=0; i<4; i++)                     // 1 pt
    for (j=0; j<4; j++)                 // 1 pt
        board[i][j] = (4*i+j+1)%16;     // 1 pt
//  Note:  There  are  other  ways,  1  pt  off  if  0  is  missed.
```

11) (10 pts) The board you initialized in the previous question is the ending position for the 15-number game, where you have a 4x4 square of numbers with one empty square. You can slide one tile from directly above, below, left or right into that empty slot for a move. (If the configuration given in question 10, you can only slide the tile from above and the left.) Fill in the function below so that it prints out the numbers of the valid tiles to move. The input to the function is the current state of the game board.

```
void printPossibleTiles(int grid[][4]) {

    int i,j, savei, savej;
    for (i=0; i<4; i++) {
        for (j=0; j<4; j++) {

            if ( grid[i][j] == 0 ) {

                savei = i ;

                savej = j;

            }
        }
    }

    if (savei - 1 >= 0) printf("%d ", grid[savei-1][savej]);

    if (savei + 1 < 4  )  printf("%d ", grid[savei+1][savej] );

    if (savej - 1 >= 0)     printf("%d ", grid[savei][savej-1]);

    if (savej + 1 < 4 )     printf("%d ", grid[savei][savej+1]);

}
```

Grading: 1 pt per blank

12) (5 pts) What is the output of the following program?

```
#include <stdio.h>
int f(int a, int b);

int main() {
    int a = 8, b = 3;
    b = f(a+b, a-b);
    printf("a = %d, b = %d\n", a, b);
}

int f(int a, int b) {
    int temp = 2*a - b;
    a = temp%17 + a;
    b = a - 2*b;
    printf("a = %d, b = %d, temp = %d\n", a, b, temp);
    return a + b + temp;
}
```

a = 11, b = 1, temp = 17
a = 8, b = 29

Grading: 1 pt each

13) (5 pts) What is the output of the following program?

```
#include <stdio.h>
int f(int* a, int* b);

int main() {
    int a = 8, b = 3;
    b = f(&b, &a);
    printf("a = %d, b = %d\n", a, b);
}

int f(int* a, int* b) {
    int temp = 2*(*a) - *b;
    *a = temp%17 + *a;
    *b = *a - 2*(*b);
    printf("a = %d, b = %d, temp = %d\n", *a, *b, temp);
    return *a + *b + temp;
}
```

a = 1, b = -15, temp = -2
a = -15, b = -16 // Grading: 1 pt each

14) (1 pt) Name one country that participated in the Russo-Japanese War. **Russia, Japan**

Fall 2011 COP 3223 Sec 3 Final Exam Free Response Solution

1) (10 pts) Write a complete program that prompts the user to enter the length and width of a rectangle in inches and then prints out both the area and perimeter of that rectangle. You are guaranteed that the user will enter positive integers less than 10000 for both items.

```
int main() {                                // 1 pt

    int length, width;                      // 1 pt printf

    printf("Enter the length and width of the rectangle.\n");
    scanf("%d%d", &length, &width);         // 1 pt

    printf("The area is %d\n", length*width); // 3 pts
    printf("The perimeter is %d\n", 2*(length+width)); // 3 pts

    return 0;                               // 1 pt
}

// Note: for each printf, 2 pts for the formula,
//                                  1    for    the    printing
```

2) (10 pts) Many students think that if a number is not divisible by 2, 3, 5 or 7, that it is prime. However, this isn't always true. For example, 121 isn't divisible by any of those four values, but 121 = 11 x 11, so it isn't prime. Arup pseudoprimes are numbers that ARE NOT prime, but also are not divisible by 2, 3, 5 or 7. Complete the program below so that it prints out all Arup pseudoprimes in between 1 and 10000, inclusive, one per line.

```
#include <stdio.h>

int main() {

    printf("Here are Arup pseudoprimes between 1 and 10000: ");
    int i;
    for (i=1; i<=10000; i++) {

        if (i%2 != 0 && i%3 != 0 && i%5 !=0 && i%7 != 0) {

            int j;
            for (j=11; j<i; j++) {
                if (i%j == 0) {
                    printf("%d\n", i);
                    break;
                }
            }
        }

    }

    return 0;
}

/*
Grading: Screening out values divisible by 2,3,5 and 7 – 3 pts
         Checking for larger divisors – 3 pts
         Printing if one larger divisor is found – 3 pts
         Floating point – 1 pt
*/
```

3) (10 pts) A common task with strings is to split them into multiple strings using some delimiters. For example, if our original string is "arup+said+to+jim,go+to+the+store" and our delimiters were the plus sign(+) and the comma(,), then the separate strings would be arup, said, to, jim, go, to, the and store. One way to carry out this split is simply to replace each of the delimiting characters in the original string with spaces. Thus, the result of splitting the string above with the delimiters + and , is "arup said to jim go to the store". Write a function that takes in a string to split and a string storing each delimiter that replaces each of the delimiting characters in the first string with spaces. Fill in the function prototype given below**. Note: You may use the strlen function.** (Hint: both strings are null character terminated!)

```
void split(char str[], char delimiters[]) {

    int i, j;                                          // 1 pt

    for (i=0; i<strlen(str); i++) {                    // 3 pts

        for (j=0; j<strlen(delimiters); j++) {         // 2 pts
            if (str[i] == delimiters[j]) {             // 2 pts
                str[i] = ' ';                          // 2 pts
                break;
            }
        }
    }
}
```

4) In this question you will write two functions that deal with a point struct that help find the area of a triangle on the Cartesian plane. The struct itself will simply store an x and y coordinate for a point on the Cartesian plane. The first function will take in two points and calculate the distance between those two points. The second function will take in three points (that define a triangle) and return the area of that triangle. If you know that the three sides of a triangle have side lengths a, b and c then the area of the triangle is $\sqrt{s(s-a)(s-b)(s-c)}$, where s represents the semiperimeter of the triangle. Thus, $s = \frac{a+b+c}{2}$. Fill in the two function prototypes provided.

```
#include <math.h>

struct pt {
    double x;
    double y;
};

double distance(const struct pt* pt1, const struct pt* pt2) {

    return sqrt(pow(pt1->x-pt2->x,2) + pow(pt1->y-pt2->y,2));

    // 1 pt for each pow call, 1 pt for sqrt call, 1 pt return
}

double triangle_area(const struct pt* ptA, const struct pt* ptB,
                     const struct pt* ptC) {

    double AB = distance(ptA, ptB);          // 1 pt
    double BC = distance(ptB, ptC);          // 1 pt
    double CA = distance(ptC, ptA);          // 1 pt

    double s = (AB + BC + CA)/2;             // 1 pt

    return sqrt(s*(s-AB)*(s-BC)*(s-CA));     // 1 pt formula,
                                             // 1 pt return
}
```

5) (10 pts) Write a function that takes in a pointer to the front of a linked list and **exchanges the values** stored in the first and last nodes of the list. (Do NOT move the location of the two nodes in the list!!!) If the list has fewer than 2 nodes, nothing should be done.

```
struct ll {
    int data;
    struct ll* next;
};

void swapFirstLast(struct ll* list) {

    if (list == NULL || list->next == NULL)  // 1 pt
        return;                              // 1 pt

    struct ll* iter = list;                  // 1 pt

    while (iter->next != NULL)               // 2 pts
        iter = iter->next;                   // 1 pt

    int temp = iter->data;                   // 1 pt
    iter->data = list->data;                 // 1 pt
    list->data = temp;                       // 1 pt

}
```