University of Maryland College Park
Dept of Computer Science
CMSC106 Fall 2013
Midterm II Key

Last Name (PRINT): _____

First Name (PRINT): _____

University Directory ID (**e.g., umcpturtle**)_____

I pledge on my honor that I have not given or received any unauthorized assistance on this examination.

Your signature: _____

Instructions

- Make sure you write your name now (we will not wait for you at the end).
- This exam is a closed-book and closed-notes exam.
- Total point value is 100 points.
- The exam is a 50 minutes exam.
- Please use a pencil to complete the exam.
- WRITE NEATLY.
- You don't need to use meaningful variable names; however, we expect good indentation.

Grader Use Only

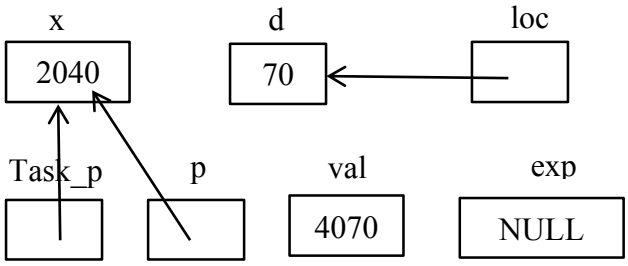
#1	Problem 1 (Memory Maps)	(14)	
#2	Problem 2 (Miscellaneous)	(26)	
#3	Problem 3 (Functions and Characters)	(30)	
#4	Problem 4 (Functions and Nested Loops)	(30)	
Total	Total (100)	(100)	

Problem #1 (14 pts)

Draw a memory map that shows the values that variables have when execution reaches the point indicated by /* **HERE** */.

```
#include <stdio.h>

void task(int *task_p, int val, int *exp) {
   *task_p += 2000;
   val += 4000;
   exp = NULL;
   /* HERE */
}

int main() {
   int x = 40, d = 70;
   int *p = &x, *loc = &d;

   task(p, d, loc);
   printf("%d %d\n", x, d);

   return 0;
}
```

Answer:



Problem #2 (26 pts)

1. (2 pts) What is the value associated with the following variable?

```
int x;
```

   a. 0
   b. We don't know (it could be any value). √
   c. 1
   d. NULL
   e. None of the above.

2. (2 pts) The following code compiles. Does it have any bugs? If so, identify them.

```
#include <stdio.h>

int main() {
   int x;
   int *p = &x;

   scanf("%d", p);
   printf("%d\n", x);

   return 0;
}
```

   Answer: No bugs.

3. (2 pts) When we pass values to function we use pass-by-value. That means:

   a. We pass copies of the values. √
   b. We pass pointers.
   c. We pass aliases of the values.
   d. We pass uninitialized data.

4. (2 pts) Local variables in a function are created when:

   a. The function returns.
   b. The function is declared.
   c. The function is called. √
   d. The function initializes them to a value other than 0.
   e. None of the above.

5. (2 pts) Assume a function has the following prototype: ***void f(int x);*** Which of the following function calls will compile?

   a. f(1.0);
   b. f(5);
   c. f(f);
   d. a. and b. √
   e. None of the above

6. (2 pts) Name two reasons why we want to use functions.

   Possible answers: Modularity, Encapsulation, Reusability, Modifiability

   Grading: Any two reasonable answers get full credit (1 pt) each. The answers don't need to be the same ones we have provided.

7. (2 pts) Does the following code compile? Yes/No answer without explanation will receive no credit.

```
#include <stdio.h>

int main() {
   int x;
   void *p = &x;

   printf("%d", *p);

   return 0;
}
```

   Answer: It does not compile. *p is not valid as it is a void pointer (needs to be int *)

8. (2 pts) Would the following be considered valid pseudocode? Yes/No answer without explanation will receive no credit.

   a. Read two values
   b. Add two values and store result in x
   c. printf("%d", x);

   Answer: It is not valid pseudocode.

9. (2 pts) What is the output of the following code fragment?

```
float x = 8.2456;
printf("%.2f %d", x, (int)x);
```

   Answer: 8.25 8

10. (2 pts) What will happen when the following code fragment is executed?

```
int *y = NULL;
*y = 5;
printf("%d\n", *y);
```

   Answer: Segmentation fault (core dump)

11. (2 pts) Parameter names don't need to be provided in a function prototype. **True**

12. (2 pts) Global variables are always initialized to 0 by default. **False**

13. (2 pts) In a function definition **int** is assumed to be the return type if no return type is specified. **True**

Problem #3 (30 pts)

Implement a function call **flip** that given a character it will change it to uppercase and vice versa. If the character provided is different than a letter, no transformation will take place. Remember that uppercase characters are in the range 65 to 90, and lowercase characters are in the range 97 to 122. To change a character to lowercase add 32 to the character value, and subtract 32 to convert to uppercase. The function will return 1 if a transformation took place and 0 otherwise. **You may NOT use functions islower nor isupper to implement this function.** The following program is an example of using the function you are expected to write. Feel free to ignore the example if you know what to implement.

```
int main() {
   char c1 = 'A', c2 = 'b', c3 = '$';
   int x1, x2, x3;
   x1 = flip(&c1);
   x2 = flip(&c2);
   x3 = flip(&c3);
   printf("c1 %c, c2 %c, c3 %c\n", c1, c2, c3);
   printf("x1 %d, x2 %d, x3 %d\n", x1, x2, x3);

   return 0;
}

void flip(char *value) {
```

```
Output:
c1 a, c2 B, c3 $
x1 1, x2 1, x3 0
```

Answer:

```
int flip(char *value) {
   /* uppercase? */
   if (*value >= 65 && *value <= 90) {
      *value += 32;
      return 1;
   } else if (*value >= 97 && *value <= 122) {
      *value -= 32;
      return 1;
   }

   return 0;
}
```

Problem #4 (30 pts)

Implement a function called **draw_triangle** that has the prototype below. The function draws a triangle where each row has one more symbol than the previous one and symbols are printed left-justified. Two examples of calling the function are provided below. Remember, your function must work for any size and for any character.

```
draw_triangle(3, '*');
```

```
*
**
***
```

draw_triangle(5, '$');

```
$
$$
$$$
$$$$
$$$$$
```

```
void draw_triangle(int size, char symbol)
```

Answer:

```
void draw_triangle(int size, char symbol) {
   int to_print = 1, row, col;
   for (row = 1; row <= size; row++) {
      for (col = 1; col <= to_print; col++) {
         printf("%c", symbol);
      }
      printf("\n");
      to_print++;
   }
}
```