University of Maryland College Park

Dept of Computer Science

CMSC106 Fall 2012

Midterm IIKey

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 200 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.

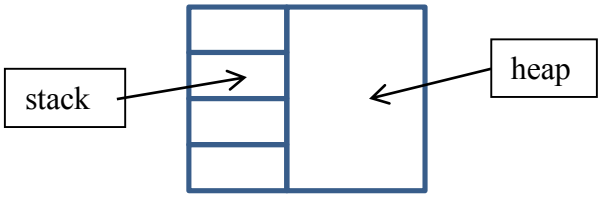
Grader Use Only

#1	Problem 1 (Memory Maps)	(20)	
#2	Problem 2 (Random Values)	(15)	
#3	Problem 3 (Function and Characters)	(15)	
#4	Problem 4 (Function andNested Loops)	(25)	
#5	Problem 5 (Function Pointer Parameters)	(25)	
Total	Total (200)	(200)	

Problem #1 (20 pts)

1. (4 pts) Draw the memory organization map discussed in lecture. Hint: it has two main components.

   Answer:



   Grading:

   The drawing must have a stack (left) and heap section (right) ((2 pts) each)

2. (16 pts) To the right of the code, draw a memory map that shows the values that variables have when execution reaches the point indicated by /* HERE */.

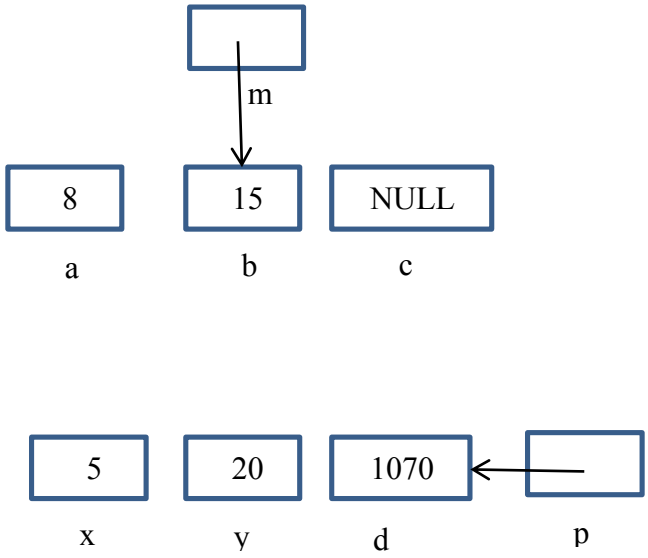
```
#include <stdio.h>

void f(int a, int b, int *c) {
   int *m = &b;
   a += 3;
   b -= 7;
   *m += 2;
   printf("%d %d %d\n", a, b, *m);
   *c += 1000;
   c = NULL;
   /* HERE */
}

int main() {
   int x = 5, y = 20, d = 70;
   int *p = &d;

   f(x, y, p);
   printf("%d %d %d\n", x, y, *p);

   return 0;
}
```



Problem #2 (15 pts)

Implement a function call **random_value** that returns a random value between **lower_limit**(inclusive) and **upper_limit**(inclusive). Remember you can get random numbers using the function **rand().**

Answer:

```
int random_value(int lower_limit, int upper_limit) {
   int range_size = (upper_limit - lower_limit) + 1;
   int value = (rand() % range_size) + lower_limit;
   return value;
}
```

Problem #3 (15 pts)

Implement a function call **find_type** that returns 1 if the parameter is an uppercase character; 2, if it is a lowercase character; and 3 for any other kind of character. Remember that uppercase characters are in the range 65 to 90, and lowercase characters are in the range 97 to 122. **You may NOT use functions islower nor isupper to implement this function.**

Answer:

```
int find_type(char ch) {
   if (ch >= 65 && ch <= 90)
      return 1;
   else if (ch >= 97 && ch <= 122)
      return 2;
   else
      return 3;
```

Problem #4 (25 pts)

Implement a function called **draw_rectangle** that generates a rectangle with the specified width and height and using the character ch. For example, calling **draw_rectangle(4, 9, '*')** will generate:

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

Remember, your function must work for any dimensions and for any character.

Answer:

```
void draw_rectangle(int width, int length, char ch) {
   int row, col;
   for (row = 0; row < width; row++) {
      for (col = 0; col < length; col++) {
         printf("%c", ch);
      }
      printf("\n");
   }
}
```

Problem #5 (25 pts)

Implement a function called **sum_and_product** that computes the sum and product of values between 1 and the **limit** value provided. The sum and product will be returned using the pointer parameters. For example:

```
int limit = 4, sum, prod;
sum_and_product(&sum, &prod, limit);
printf("sum: %d, prod: %d\n", sum, prod);
```

will generate the output:

```
sum: 10, prod: 24
```

Answer:

```
void sum_and_product(int *sum, int *prod, int limit) {
   int i;

   *sum = 0;
   *prod = 1;
   for (i = 1; i <= limit; i++) {
      *sum += i;
      *prod *= i;
   }
}
```