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

<table>
<thead>
<tr>
<th>#1</th>
<th>Problem 1 (Memory Maps)</th>
<th>(20)</th>
</tr>
</thead>
<tbody>
<tr>
<td>#2</td>
<td>Problem 2 (Random Values)</td>
<td>(15)</td>
</tr>
<tr>
<td>#3</td>
<td>Problem 3 (Function and Characters)</td>
<td>(15)</td>
</tr>
<tr>
<td>#4</td>
<td>Problem 4 (Function and Nested Loops)</td>
<td>(25)</td>
</tr>
<tr>
<td>#5</td>
<td>Problem 5 (Function Pointer Parameters)</td>
<td>(25)</td>
</tr>
<tr>
<td>Total</td>
<td>Total (200)</td>
<td>(200)</td>
</tr>
</tbody>
</table>

Total (200)
Problem #1 (20 pts)

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

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 */.

```c
#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\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()`.

```c
int random_value(int lower_limit, int upper_limit)
```
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.**

```c
int find_type(char ch)
```
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.

```c
void draw_rectangle(int width, int length, char ch)
```
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:

```c
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
```

```c
void sum_and_product(int *sum, int *prod, int limit)
```