2016 Fall COP 3223H Final Exam 12/7/2016

Name: _____

The game of Uno consists of cards that have a color and a number. The four possible colors are: red, green, blue and yellow and the numbers are one through nine, inclusive. (A real Uno deck has some other special cards, but to keep these questions easier we are only modeling the "regular" cards in the deck.) For the first X questions, please assume the file to which you are adding functions starts as follows:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>

#define FULLDECK 72
#define NUMCOLORS 4
#define MAX 20
const char COLORS[NUMCOLORS][MAX] = {"red", "green", "blue",
"yellow"};

typedef struct UnoCard {
    char color[MAX];
    int number;
} UnoCard;

typedef struct UnoDeck {
    UnoCard cards[FULLDECK];
    int numCards;
} UnoDeck;
```

1) (8 pts) Write a function, randCard that takes in a pointer, ptr, to a UnoCard and fills the card with a random color and number in the ranges described above. You may assume that the random number generator has already been seeded and that you may call the rand() function as needed as well as any string functions.

```
void randCard(UnoCard* ptr) {









}
```

2) (5 pts) Write a function, printCard that takes in a const pointer, ptr, to a UniCard and prints it out. A green 7 should simply print out "green 7".

```
void printCard(const UnoCard* ptr) {



}
```

3) (8 pts) Write a function, isMatch, that takes two constant pointers to 2 UnoCards and returns 1 if they are a match and 0 otherwise. Two UnoCards are matches if and only if they either share the same number, the same color, or both.

```
int isMatch(const UnoCard* ptr1, const UnoCard* ptr2) {









}
```

4) (20 pts) Write a function, fillDeck, that takes in a pointer to a single UnoDeck, and fills it with 72 cards, 2 of each possible type. In particular, the cards should be ordered red 1, red 2, …, red 9, green 1, green 2, …, green 9, blue 1, …, blue 9, yellow 1, …, yellow 9, red 1, red 2, …, yellow 9. (Thus, fill the first 36 cards as specified, sorted by color then number, then put a second copy of each card in that same order.) Don't forget to set the number of cards in the deck to 72.

```
void fillDeck(UnoDeck* ptr) {









}
```

5) (10 pts) Write a function, shuffleDeck, that takes in a pointer to a single UnoDeck, and shuffles it. To shuffle, simply choose two random indexes and swap their contents 1000 times. You may call the swapUnoCards function that is described below:

```
// Swaps the contents of UnoCards pointed to by ptr1 and ptr2.
void swapUnoCards(UnoCard* ptr1, UnoCard* ptr2);

void shuffleDeck(UnoDeck* ptr) {









}
```

6) (15 pts) Write a drawCard function that takes in a pointer to a single UnoDeck and returns an UnoCard struct. The function should choose a card at random from the deck, copy its contents into a temporary UnoCard variable, copy the contents of the last card in the deck to the location of the removed card, reduce the number of cards in the UnoDeck and then return the temporary UnoCard variable storing the card drawn. (A function can return a struct in this manner. Typically, the function calling it will assign the struct returned to a struct defined in the calling function and the contents will properly get copied.)

```
UnoCard drawCard(UnoDeck* ptr) {









}
```

7) (10 pts) Jordan runs 2 miles every weekday, 5 miles every Saturday and 6 miles every Sunday. Write a complete program ***without*** a loop that prompts the user to enter the number of days for a time interval starting on Monday, reads in this value, and prints out the total number of miles Jordan has run. (Hint: Each complete week, Jordan runs 21 miles.) For example, for a time interval of 30 days, he's run 21 x 4 + 4 = 88 miles since he's run for four complete weeks and 2 leftover weekdays (Monday and Tuesday).

```
#include <stdio.h>

#define REG_DAY 2
#define SATURDAY 5
#define SUNDAY 6
#define WEEK (5*REG_DAY+SATURDAY+SUNDAY)

int main() {









}
```

8) (10 pts) Define the "power tower" function, pt, as follows:

$$pt(1) = 1,\ pt(n) = n^{pt(n-1)},\ \text{for all integers } n > 1.$$

Write a **recursive** function that takes in n and returns pt(n). (Note: In practice your function can only be called with 1, 2, 3 or 4. Also, the exponential function in math.h is pow.)

```
int powertower(int n) {







}
```

9) (12 pts) Here is an example of using Horner's Method to evaluate a polynomial:

Let $f(x) = 3x^4 - 2x^3 + x^2 + 5x - 6$

Consider evaluating f(2):

We first set our result to 0.

We multiply our current result by the value of x (in this case 2) and then add our current coefficient (reading most significant to least significant), 3, to it to obtain 3.

We repeat, taking the current result (3) and multiplying it by x (2) and adding the next coefficient, -2 to obtain 4.

Then we take 4, multiply by 2 and add to it 1 (the next coefficient) to obtain 9.

Then we take 9, multiply by 2 and add to it 5 (the next coefficient) to obtain 23.

Finally we take 23 multiply it by 2 and add -6 (the last coefficient) to get a final answer of 40.

Indeed, $f(2) = 3(2)^4 - 2(2)^3 + (2)^2 + 5(2) - 6 = 48 - 16 + 4 + 10 - 6 = 40$.

Write a function that takes in an integer array storing the coefficients of a polynomial (f[i] stores the coefficient of the term x^i), the length of the array, and an integer x at which to evaluate the function and returns the value of the function f evaluated at x, using the method described above. (You will be graded on how closely you implement the algorithm described above instead of the correctness of your code. Namely, an incorrect algorithm that comes close to follow these steps will get more credit than a correct algorithm that solves the problem in a different way. I am testing your ability to read a description of an algorithm that has a loop and convert it to code.)

```
int eval(int f[], int x, int length) {




}
```

10) What's the primary color on the Beattles Album, Yellow Submarine? _____

Scratch Page - Please clearly mark any work on this page you would like graded.