Midterm 1 Practice

All code should be written in C++. Unless otherwise specified, you may (and I generally will):

- assume that the user of any code you write will be cooperative with the input they supply;
- omit `std::` and the return value of `main()`;
- assume that all necessary libraries have been `#include`d;
- omit `main()` entirely for problems that ask ONLY for function definitions;
- not concern yourself with having optimal solutions (within reason);
- not worry yourself about prompt messages for user input (I sometimes give descriptive prompts to clarify problems);
- not recopy code I have provided, or which you have written in other parts of problems.

1. Write a program that allows the user to enter a time span in the format hours:minutes:seconds, e.g. `2:11:03`. The program should print out the number of seconds that this time period would represent (in the example given, that would be 7863 seconds).

   [Note: if the user types in `03` at a `cin` prompt which is waiting for an `int`, then the value `3` will successfully be stored to that `int`.]

2. Write code which allows the user to enter a `string`, which you may assume consists entirely of letters, upper or lowercase. Write code, without using any imported function like `std::toupper()`, which prints out the string, but with all letters changed to uppercase. (Hint: even if you don't remember ASCII values, there is a simple numerical trick to derive the offset.)

3. When I run the following program, it starts off displaying `2147483600`, `2147483601`, `2147483602`, like you would expect. But then it starts displaying negative values. Explain why that would happen.

```
int main(){
    int x = 2147483600;
    for(int i = 0; i <= 1000; ++i) {
        std::cout << x + i << std::endl;
    }
}
```

4. Identify the ssssssssttttttuuuupppppiiiddd mistake that would cause a compiler error. (Hint: anyone who is coming from a Python background will likely make this mistake at some point. Look carefully.)

```
#include <iostream>
#include <string>

int main() {
   int x = 5;
   int y;
   double z = 4;
   std::string pq = 'Hello';
   std::cout << x << y << z << pq;
}
```

5. a. The following program has one line that will cause a compiler error. Identify the line, and remove it.

   b. What will the following program display?

```
void my_f(int &b, int c) {
   b += c;
   c += b;
   cout << "1)" << b << " " << c << endl;
   return b;
}

int main() {
   int a = 5, b = 20;
   cout << "2)" << a << " " << b << endl;
   my_f(a, b);
   cout << "3)" << a << " " << b << endl;
}
```

6. Write code which asks the user ten times to enter an integer (one entry per line). If the user ever enters `7` or `11`, the program should immediately stop asking and print `WIN`. Otherwise, if the user never enters those values, the program should print `LOSE`.

7. Write code which asks the user to enter a positive odd integer N, and then prints the following $N \times N$ grid, where the top left corner is a star, and there is a chessboard pattern (shown here with $N = 11$):

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

8. A positive integer is called *perfect* if it is equal to the sum of its factors (aside from the number itself). For example, 28 is perfect, because the factors of 28 are 1,2, 4, 7, and 14, and $28 = 1 + 2 + 4 + 7 + 14$. 6 is another perfect number, since the factors of 6 are 1, 2 and 3, and $6 = 1 + 2 + 3$.

   Write code which (given enough time) prints out the value of the seventh perfect number.

   As part of your solution, write at least one function (aside from `main()`), and call that function from `main()`. The function you write should return a value, and not contain any `cout` statements.

9. Write code that allows the user to enter exactly 10 names. After that is done, the program should print out the names in reverse order.

10. Write code that creates a 10×10 array of `double`s, with each entry initialized to a random value between 0.0 and 1.0. The program should then print out the greatest entry in every "row" (i.e., `x[i][j]` would be an entry in the `i`th row).

    You may assume that the code

```
std::random_device rand_dev;
std::mt19937 gen( rand_dev() );
std::uniform_real_distribution<> dist(0.0, 1.0);
```

    already appears in your code; and recall that `dist(gen)` will produce a random value from the distribution.

11. What will the following program display?

```
int main() {
   int x[6] = {10, 15, 20, 25, 30, 35};
   cout << x + 3 << " " << &(x[1]) << " " << *x + 4 << " " << *(x + 4);
}
```

    If the address of the first element of `x` is `0x50528c`, what would you expect the rest of the outputs to be?

12. Write a recursive function named `num_digits()` which receives a positive `int` argument, and returns the number of digits it has. Do not use any string conversion techniques, nor any loops.

13. Write a recursive function named `is_palin()` which returns a `bool` indicating whether or not a string is a palindrome. Your function should receive a `string` as argument, and could potentially receive other arguments as well. Your function should not utilize any loops.

14. Every positive integer has a *prime factorization*, which is unique (aside from reordering the factors). For example, 168 can be written as $2 \cdot 2 \cdot 2 \cdot 3 \cdot 7$, where each factor is prime. This factorization has five factors.

    Write a RECURSIVE function called `num_p_factors` which receives a positive `int` as argument, and returns the length of its prime factorization. For example, `num_p_factors(168)` should return `5`. (And also, `num_p_factors(1)` should return `0`.)

    Your function can use a loop, but it should not use any nested loops.

15. Write a recursive function `addstar()` which receives a `string` as argument. The function should return the same `string`, except with asterisks between each pair of characters in the input. For example, `addstar("world")` should return `"w*o*r*l*d"`. Your function should contain no loops. However, your function can use the *substring* function: if `x` is a `string`, and `i` is an `int`, then `x.substr(i)` returns the contents of `x` from the character with index `i` until the end.

16. Write a recursive function named `dubdub()`. Your function should receive a `string` `x` as argument, and could potentially receive other arguments as well. Your function should return a `bool`, which is `true` if the string `x` contains the same two-letter substring at (at least) two different locations. For example, `banana` contains `an` at two different locations, and `herder` contains `er` at two different locations. Your function should not utilize any NESTED loops, although it can contain an unnested loop.

17. a. What will the following program display?

```
int* fn(int *ptr_arg, int val_arg) {
   *ptr_arg += val_arg;
   int *new_ptr = new int; // **Line 1
   *new_ptr = val_arg;    // **Line 2
   return new_ptr;
}

int main() {
   int x = 20, y = 78, *p, *q;
   p = &x;
   q = fn(p, y);
   cout << x << " " << y << " " << *p << " " << *q;
}
```

b. Imagine that the two lines marked `**Line1` and `**Line2` were removed, and replaced with the single line

```
int *new_ptr = &val_arg;
```

Explain why this could lead to unpredictable print outs.

18. Write a function named `repeat` which receives one `int` array named `x` and an `int` argument `length`, and returns an `int` pointer. The return value should point to an array that is twice the length of `x`, which contains `x` concatenated with itself: that is, the first `length` entries of the new array will just be the entries of `x`, and the last `length` entries of the return will just be `x` again.

For example, the following code should print `1 4 9 1 4 9`:

```
int x[] = {1,4,9};
int *y = repeat(x, 3);
for(int i = 0; i<6; ++i){
    cout << y[i] << " ";
}
```

19. Consider the following program:

```
int main() {
    string names[] = {"alice", "joe", "andrew", "bob", "carol", "charles"};
    int length = 6;
    char **table = new char*[length];
}
```

I wish to have each entry of `table` point to an array of `char`s which contains the letters from the corresponding `string`. For example, with the given contents of `names`, the first entry of `table` should point to an array of `char`s of length 5, containing `'a', 'l', 'i', 'c', 'e'` as entries; the second entry should point an array of `char`s of length 3 containing `'j', 'o', 'e'`, etc.

Write code that fills out `table` as described, which works even if `names` was initialized with different values (but such that `length` still is the length of `names`).

At the end, also write code that deallocates all dynamically allocated memory.

20. I have a file named `script.txt` which contains several lines of dialogue for a play with several play-characters. For example, the file could look like

```
Alice: Hello, my name is Alice. Who are you?
Bob: I'm Bob, and I'm the main character in this play.
Alice: I disagree, I am the main character. I have the most lines.
Bob: I think we have an equal number of lines.
Charlie: What about me? I have a line too!
```

Each line of the file should start with the (single-word) name of a play-character, followed by a space, followed by a `:` and some dialogue.

Write code which prints out the name of each play-character who appears in the play. It is ok if a character's name appears several times.

21. Write a program which reads through the text file from the last problem, and prints out the LONGEST line (the line with the most characters).

22. Consider the following code.

```
int main() {
   int x = 23, y = 5;
   string s;
   char c;
   cin >> x >> y >> s >> c;
   cout << x << " " << y << " " << s << " " << c;
}
```

What would happen if the user entered each of the following? Be as precise as possible.

a. `102 304 hello world goodbye`

b. `a b c d`

c. `12 34hijk.`

23. Write a program that opens a text file named `script.txt`, and then creates a new text file called `tpircs.txt`, by taking each line of `script.txt` and writing the words in reverse order. (A "word" should be considered to be a run of consecutive, non-whitespace characters.) For example, with the file from Problem 20, the contents of the new file would be

```
you? are Who Alice. is name my Hello, Alice:
play. this in character main the I'm and Bob, I'm Bob:
lines. most the have I character. main the am I disagree, I Alice:
lines. of number equal an have we think I Bob:
too! line a have I me? about What Charlie:
```

Accomplish this WITHOUT using arrays (or vectors) of any sort. Instead, utilize string concatenation and stringstreams.