# Physics 2200 Midterm I Project

- The goal of the programming project is to implement in C a recursive algorithm for Coin Change Problem and investigate the performance of your implementation.

  In coin change problem, you have to make a certain sum using given coin denominations. Let's number the coins from 1 to N and the sum you are trying to get S. The problem is basically finding in how many ways we can make the sum S using coins that have numbers from 1 to N.

- *Reproducible research* is a concept relevant for data analysis or computational projects. It requires that the data and the computer code which went into doing that analysis be available to others so that they might examine what you've done and reproduce your findings.

  In the spirit of reproducible research approach, your code must be 'compilable' (using make) and 'runnable' and produce the results without any grader's intervention.

- You are welcome to discuss the project's ideas with others in order to better understand them but the work you turn in must be your own. In particular, you must write your own code, run your own calculations, and communicate and explain the results in your own words.

Name: _____

Date: _____

| Question: | 1 | 2 | 3 | 4 | 5 | 6 | Total |
|---|---|---|---|---|---|---|---|
| Points: | 5 | 15 | 5 | 5 | 5 | 10 | 45 |
| Score: | | | | | | | |

**Suggested project steps:**

1. (5 points) Create a GitLab repository for your project, call it **m1**. As you develop your code, upload there all your C code, your Makefile, and .gitignore file.

2. (15 points) Write a C function with the following declaration

```
int count (int *v, int nv, int amount, int i)
```

   that counts the number of ways the change in amount of **amount** can be given using the set of coin denominations stored in an integer array of the size **nv**.

3. (5 points) Write a C program that loops over the value of the parameter **amount** mentioned above, from $amount = 1$ to $amount = 100$. At each iteration print the amount and the number of ways to give the change. Chose the coins as following:

```
int v[] = {1, 2, 3, 4, 5};
nv = sizeof(v)/sizeof(*v);
```

   Do not forget to check in the working version of your code to GitLab.

4. (5 points) Continue expanding your program by adding code that measures the time per function call. To increase the precision of your timing, repeat functions calls multiple times such that the total running times for each value of **amount** is between $tmin = .1$ and $tmax = .2$ seconds.

```
count = 1000; // some number
do
{
    timer_start ();

    // place your code to time here

    time = timer_stop ();
    time1 = time / count;
    printf (" %10.2f usec     %10.6f sec    %10d\n",
        time1 * 1.e6, time, count);
    /*
     * adjust count such that cpu time is between
     * tmin and tmax
     */
    count = adjust_rep_count (count, time, tmin, tmax);
}
while ((time > tmax) || (time < tmin));
```

The code for timing and adjusting the value of count is provided on the class website.

The code for timing is provided on the class website.

5. (5 points) Plot a graph of the time per function call (time1 variable above) versus the amount variable. Use a scripting program, e.g. gnuplot.
6. (10 points) Your C code should be well commented, written elegantly, and properly and consistently formatted. Your repository shoud contain snapshots of your code as you develop it.