# Project 1 – Numbers

## Objectives:

In this project, you will gain experience in developing C programs from multiple source files in a Unix environment and compiling them using makefiles.

## Requirements:

For this project, you are required to develop a command line program that depending on command line arguments creates a randomized array, reverses each element of that array, & compares the two arrays for similarity.

Project structure requirements are as below:

1. The name of your output program should be proj.

2. The program will accept exactly 1 argument & should run as follows:
   a. **$ ./proj 1**
      This will print the array of randomized numbers.
   b. **$ ./proj 2**
      This will print the reversed & original form of randomized numbers.
   c. **$ ./proj 3**
      This will print similar words & total no of similar words found in both arrays.

3. The project consists of 3 parts. Complete each part before going onto the next part as each subsequent part is built upon the one before.

4. You have been provided a skeleton project to which you may add your code to complete the project. The project in its current state compiles and prints a usage message. You have been provided with the following files:

   a. **makefile**: The provided makefile can be used to compile all three source files to ease compilation. You can use it in the following way
      $make
      This will compile the source code & generate ‘proj’ provided it is error free
   b. **proj1.c**: This file should contain the ‘main()’ function of your program.it will be used to call the functions implemented in proj1_functions.c

c. **proj1_functions.c**: This file should contain all the source code (i.e. functions, global variable declarations, etc.) dealing with operations.
d. **proj1_header.h**: This file should contain the function prototypes of all the functions in proj1_functions.c that need to be called from any other file (e.g. the proj1.c file).
e. **proj1_ref:** My implementation so you can run & see how the output of your program should look like.
f. If you have confusions you can look up the **sample project** I have included here.

The requirements for each part of the project are as below:

- **Step One :**
  Generate a randomized array of a 100 numbers between the range of 1 & 1000.
  You can get random numbers with the help of functions rand() & srand().
  **Sample Output :**
  ```
  $ ./proj 1
  Random Array
   383   383   290   746   301    675   541   677   896   915
   295   423   760   890   683     93   444   265   598   449
   911    56   170   902   873     61   557   517   131   855
   533   896   178   303   358     64   156   199   320   636
   306   445   166   572   249    290   389   790   567   692
  ```

- **Step Two :**
  Take the generated array & reverse the value of each index like a palindrome & save it in a separate array.
  for example ,
  10 becomes 1 , 100 becomes 1, 789 becomes 987 & so on.
  **Sample Output :**
  ```
  Random Arrray print .. . .
  Reversed
  383   383    92   647   103   576   145   776   698   519
  592   324    67    98   386    39   444   562   895   944
  119    65    71   209   378    16   755   715   131   558
  219   335   698   871   303   853    46   651   991    23
  913   636   603   544   661   275   942    92   983    97
  ```

- **Step Three:**
  Take the array from the previous step & compare it to the randomized array like a linear search .
  foreach equal value found print the value & count the total no of similar values found.

  Sample Output :

```
Random Array print . . .
Reversed Array print . . .
Similar Numbers
383    383    383    383    915     93    444    131    533    303
636    202    474    515     39      4    222    519     11    335
 737    414
Total no of reversed words found are 22
```

**NOTE :**

1. When grading your program, only valid integers will be used as input, so there is no need to verify if program arguments are integers. However, the number of nodes or the number of lists inputs maybe be zero, so check for this case and display the correct output.
2. Your grade on this project will depend upon passing tests cases. This means your program's output should EXACTLY match the output of the reference implementation. To facilitate your testing, you will be provided with sample outputs and test scripts to ensure that you output is as required

**Grading Criteria:**

Your grade on this assignment will depend upon:

1. This project is worth 20% of your Total Project Marks.
2. Following the coding standards (posted on GitHub).
3. Fulfilling project requirements.
4. Performing the required processing correctly and displaying the correct results
5. Handling all corner cases correctly
6. If your project does not compile, it's an AUTOMATIC ZERO, no exceptions
7. Performing validation checks on your input. There should be no crashes or incorrect results due to improper input. However, for any numeric arguments, you can always assume that your program will be only tested with numbers, NOT any other non-numeric input
8. During testing, your program should not crash; marks will be deducted for any crashes during testing.

**Tips:**

- Do not try to implement everything in one go, do it step by step e.g. first allocate the required memory, then initialize it, then set up the array . Test at each step to ensure your program is running correctly.
- Complete one part of the project before going on to the next part.
- You will get partial credit, so always ensure you have a running program at each step, even if you cannot complete the entire project
- Do not leave everything for the last minute, start early so that you have enough time to implement, test, debug and ask for help if needed.
- Be careful when using pointers, understand what you are doing and test often.
- If you are stuck, post a question on Google Classroom.
- If you work carefully, this is an easy project, which should only take you 2-3 days. Good Luck!