1 Objectives

The objectives of this lab are:

1. Gaining more practice with C in general.
2. Gaining more practice with writing and calling your own functions.
3. Getting more practice with memory allocation and deallocation.
4. Gaining some experience with detecting integer overflow.
5. Learning to change text color (if you have not done so already).

2 Requirements

Big picture: You will be filling up a dynamically allocated array with potentially large random numbers. You will then use different approaches to add up the numbers in the array.

2.1 Program Description/Requirements

This program generates an array of random numbers. After the array is allocated and filled with numbers, a function shall be called to display each element of the array, as well as the subtotal up to that point.

- The first column of the output will display each random number.
- The second column shall display the running total **with** overflow detection in an `unsigned int`.
- The third column will display the subtotal **without** rollover detection in an `unsigned int`.
- The fourth column will display the subtotal without rollover detection in an `unsigned long long`.

For example, assume a 3-element array was populated as shown below:

49066129
87851954
7065299

In this case, the output would look like the following:

```
                  Unsigned       unsigned       unsigned
                       int            int      long long
                  Subtotal       Subtotal       Subtotal
       Random    w/ Rollover   w/o Rollover   w/o Rollover
       Number      Detection      Detection      Detection
-------------  -------------  -------------  -------------
     49066129       49066129       49066129       49066129
     87851954      136918083      136918083      136918083
      7065299      143983382      143983382      143983382
```

Each column shall be right-justified as shown in the example to allow for a 13-digit number.

When calculating the subtotal for the second column, if adding a number to the current total would cause a rollover, then the output shall signify that. When printing the output, it shall look like the following (including the red text):

[prior numbers not shown]

```
      3163711     3959034731     3959034731     3959034731
     76146772     4035181503     4035181503     4035181503
     29027890     4064209393     4064209393     4064209393
    115020362     4179229755     4179229755     4179229755
    100109211     4279338966     4279338966     4279338966
    123456789       REJECTED      107828459     4402795755
    100109211       REJECTED      207937670     4502904966
            3     4279338969      207937673     4502904969
    109822771       REJECTED      317760444     4612727740
```

[continue the display of all numbers]

2.2 Additional Requirements and Details

1. The name of the code file shall be “main.c”.
2. The number of elements of the array is also chosen randomly, from a minimum size of 1 element up to a maximum size of 100 elements.
3. Each element of the array shall be an “unsigned int”.
4. The variables holding the running totals for the second and third columns shall be declared as “unsigned int”, while the variable holding the running total for the fourth column shall be declared as “unsigned long long”.
5. The array must be allocated dynamically using `malloc`.
6. Limit the maximum size of each random number stored in the array to a number less than 1,234,567,890.
7. Do not use global variables.
8. You shall call a function called `display` that will display the results on the screen as described. Only one call to this function is allowed. The function prototype shall be:
   ```
   void display(unsigned int *values, unsigned int num_values);
   ```
9. Free the memory you have used before the program terminates.

3 Submission

Submit main.c to Sakai by the deadline.

4 Grading

Your grade shall be based on the following **guidelines**:

1	Compiles without errors:	30	
2	Compiles without warnings:	10	
3	Proper formatting of output:	5	
4	Valgrind reports no errors	15	(includes no segmentation faults)
5	Valid output	15	
6	Style guide followed properly	10	
7	All code requirements met	15	(malloc, display, etc.)

I reserve the right to deduct points when it appears appropriate.