Checksum.h

```c
/* Checksum.h - INTERFACE: Computes the 8 or 16 bit checksum of a buffer
 * By: Stephane Durette
 * Date: Sept 16 2019
*/

//#ifndef CHECKSUM_H
//#define CHECKSUM_H

// Define enumerated data type
typedef enum { CHK_8BIT, CHK_16BIT, CHK_ERROR } CHECKSUM;

// Function Prototype
unsigned int Checksum(void* buf, int iBufLen, CHECKSUM iType);

                                        //#endif
```

Checksum.c

```c
/* Checksum.c - IMPLEMENTATION: Computes the 8 or 16 bit checksum of a buffer
 * By: Stephane Durette
 * Date: Sept 16 2019
*/

#include "Checksum.h"
#include <stdio.h>

unsigned int Checksum(void* buf, int iBufLen, CHECKSUM iType) {
	unsigned short* newBuf16; //iType == CHK_16BIT
	unsigned char* newBuf8;     //iType == CHK_8BIT

	unsigned int sum = 0, checksum = 0;

	switch (iType) {
	case CHK_8BIT:
		newBuf8 = (unsigned char*)buf; //cast buf to char
		for (int i = 0; i < iBufLen; i++) {
			sum += newBuf8[i];
		}
		checksum = sum % 0x100;
		return checksum;
	case CHK_16BIT:
		newBuf16 = (unsigned short*)buf; // cast buf to short
		for (int i = 0; i < iBufLen; i++) {
			sum += newBuf16[i];
		}
		checksum = sum % 0x10000;
		return checksum;
	default:
		//signal an error
		printf("\nYou must select either * bit or 16 bit checksums");
			return 0x100000;
	}
                                        }
```

Main.c

```c
/* main.cpp : Testing mainline for Lab #3
*            By: Michael Galle & Jack Cole
*/

#include <stdio.h>
#include <string.h>
#include <malloc.h>
#include <stdlib.h>
#include "Checksum.h"       // Header file includes Prototype defined in Checksum.c

int main(int argc, char* argv[])
{
	int                                i, rc, iPackedLen;
	unsigned char* bPackedString;
	unsigned char        bZeros[6] = { 0 };
	int                                iBit;
	unsigned char        iMask;

	// Test the Checksum() function - 'Checksum' (LAB 3)

	// First checksum a 0 buffer (both 8 and 16 bit cases)
	i = Checksum((void*)bZeros, 6, CHK_8BIT);
	printf("\n\n8 bit checksum of 0's is %d.\n", i);
	i = Checksum((void*)bZeros, 3, CHK_16BIT);
	printf("16 bit checksum of 0's is %d.\n", i);

	// Checksum a random buffer (non-zeros --> need malloc() since calloc() sets all
to zeros and don't need to do this )
	bPackedString = (unsigned char*)malloc(1024);
	// void *malloc(int size)   --> size = size of memory block in bytes, returns
pointer to start of memory block or NULL if request fails
	for (i = 0; i < 1024; ++i) {
		bPackedString[i] = rand() & 0xff;
			// Generate a random single byte numbers and place in allocated
memory
	}
	rc = Checksum((void*)bPackedString, 1024, CHK_8BIT);
	printf("\n 8 bit checksum of buffer is %d.\n", rc);

	// Zap some bits (change it) and report change in checksum

	// get a byte to flip. We'll just zap some middle bit
	rc = rand() % 1024;
                                    // Choose a number between 0 and 1024 (there are 1024
bytes to choose from that can be altered)
	iBit = rand() % 8;
                                    // Choose a number between 0 and 8 (one of the 8 bits
in the chosen byte)
	iMask = 1 << iBit;
                                    // Left shift 0000 0001 by a random number of bits
between 0 and 8 (to select bit)
	if (bPackedString[rc] & iMask) bPackedString[rc] &= ~iMask;
	// If selected bit is a 1 then invert the mask and 'and' bPackedString[] with the
inverted mask --> this flips only the selected bit
```

```
        else bPackedString[rc] |= iMask;
                        // If selected bit is 0 then keep mask as it is and 'or'
bPackedString[] with mask --> this flips only the selected bit
        rc = Checksum((void*)bPackedString, 1024, CHK_8BIT);
        printf("8 bit checksum of altered (by 0x%x) buffer is %d.\n", iMask, rc);
        free(bPackedString);

        printf("Press any key to end program ...");
        getchar();  // Keep console open until user presses enter
        return(0);
}
```

Program output