University of California, San Diego
Dept. of Computer Science and Engineering
CSE 30 – Computer Organization and Systems Programming
Problem Set #2

Problem 1: Overflow

Consider the following binary arithmetic operations:

```
101101 + 001000
110100 - 101100
000100 + 111101
```

a) Which of the following 6-bit, ***unsigned*** integer arithmetic calculations cause *overflow*?
b) Which of the following 6-bit, ***two's complement*** integer arithmetic calculations cause *overflow*?

Problem 2: String Manipulation

Write a function takes as arguments a string and a target character and returns the index of the last occurrence of that character in the string. The function returns -1 if the character is not found. The method header is given below.

int cse30strrchr(const char * s, char target);

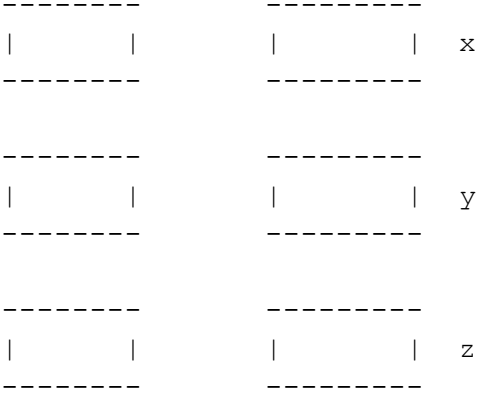
Problem 3: Pointers

Using the diagram next to the code for the program variables, complete the diagram to show the values of each of the variables when the program is finished. Use decimal values for the integers and arrows for the pointers.

```
int main(int argc, char *argv[]) {
   int x;
   int y;
   int z;
   int *px;
   int *py;
   int *pz;

   px = &x;
   py = &y;
   pz = &z;
   x = -1;
   y = 14;
   z = 20;

   y = *px + 10;
   *pz = *px;
```



```
    *px = *py + z;
    py = pz;
}
```

Problem 4: Fibonacci Errors

Consider the following code which should prints out a table of Fibonacci numbers. These are defined by a series in which any element is the sum of the previous two elements. This program stores the series in an array, and after calculating it, prints the numbers out as a table. However, this does not print out the right values. What codes needs to be modified or added in order to make it work as intended?

```
#include <stdio.h>

main()
{
      int fib[24];
      int i;

      for(i = 2; i < 24; i++)
            fib[i] = fib[i-1] + fib[i-2];

      for (i = 0; i < 24; i++)
            printf("%3d   %6d\n", i, fib[i]);
}
```

Problem 5: **Average Errors**

This function takes in an integer array arr and an integer n, the size of the array. Then it will calculate the average of the elements in the array. There are 6 problems with this function. Locate the problems and then rewrite the function so that it will work properly.

```
void average (int [] arr, int n)
{
    for (i = 0; i ≤ n; i++)
    {
        sum = arr[i] + sum;
    }
    int result = sum/n;
    printf ("The average is: %lf" result);
}
```

Problem 6: **Nested Loops**

Consider the following code:

```
void filter(int image[], int result[], int N, int M)
```

```
{
	int n;
	int m;
	for(m = 1; m < M - 1; ++m){
		for (n = 1; n < N - 1; ++n){
			result[m*N+n] = (image[(m - 1) * N + n - 1] +
				image[(m - 1) * N + n] +
				image[(m - 1) * N + n + 1] +
				image[m * N + n - 1] +
				image[m * N + n] +
				image[m * N + n + 1] +
				image[(m + 1) * N + n - 1] +
				image[(m + 1) * N + n] +
				image[(m + 1) * N + n + 1]) / 9;
		}
	}
}

main() {
	int N=5;
	int M=5;
	int i;
	int image_input[25];
	int result_output[25];

	for (i = 0; i < M*N; ++i){
		image_input[i] = i;
		result_output[i]  = 0;
	}

	filter(image_input, result_output, N, M);
}
```

a) What are the values in `result_output` array?
b) What are the values in the `result_output` if you change `result[m*N+n]` to `result[(m - 1) * (N - 2) + n - 1]` in the `filter` function ?

Problem 7: Binary Manipulation
Write a C function that takes a binary in unsigned int form and returns how many 1s are there in the odd digit of the binary. For example, if giving an input of 0b11001001, it should return 2. And 0b01010101 returns 4.

The function header is:
```
int oddBitCount(unsigned int binary);
```