CSC 212: Data Structures and Abstractions
University of Rhode Island
Fall 2018
Weekly Problem Set #1

This assignment is due Thursday 9/20 before lecture. Please turn in neat, and organized, answers hand-written on standard-sized paper **without any fringe**. The only library you're allowed to use in your answers is `iostream`, though you can test with whatever you'd like. Problem set 1 is about strings, arrays, functions, and pointers, some of the most fundamental concepts in C/C++ programming.

1. Provide a sequence of Bash commands that will:

   - go to your default home directory;
   - create a directory test;
   - rename test to myproject;
   - enter the directory myproject;
   - create a new empty file main.c;
   - list all files in myproject, including hidden files;
   - return to the parent directory.

   Solution:

```
cd
mkdir test
mv test myproject
cd myproject
touch main.c
ls -a
cd ..
```

2. Provide a sequence of Bash commands that will:

   - create files a.txt, b.txt, and c.txt;
   - write the line a: 1 2 3 4 5 to a.txt;
   - write the line b: 6 7 8 9 10 to b.txt;
   - write the line a: 11 12 13 14 15 to c.txt;

- concatenate a.txt, b.txt, and c.txt into all.txt.

Solution:

```
touch a.txt b.txt c.txt
echo "1 2 3 4 5" >> a.txt
echo "6 7 8 9 10" >> b.txt
echo "11 12 13 14 15" >> c.txt
cat a.txt b.txt c.txt > all.txt
```

3. Write a function that returns the length of a given string. For example, given `"Test"`, return `4`.

   Solution:

```
unsigned length(char* input)
{
    unsigned length = 0;
    while(*input)
    {
        input++;
        length++;
    }
    return length;
}
```

4. Write a function that returns the number of words in a given string. Words are always separated by whitespace or tab characters.

   Solution:

```
unsigned wordNum(char* input)
{
    unsigned wordNum = 0;
    bool isWord = false;
    while(*input)
    {
        if(isWord)
        {
            if(*input == 32 || *input == 9)
            {
                isWord = false;
```

```
                wordNum++;
            }
        }
        else
        {
            if(*input != 32 && *input != 9)
                isWord = true;
        }
        input++;
    }
    if(isWord)
        wordNum++;
    return wordNum;
}
```

5. Write a function that returns a missing number in an array of integers ranging from 1 to n. For example, given $[3, 2, 1, 5]$ and $n = 5$, output 4.

   Solution:

```
unsigned missing(unsigned* input, unsigned n)
{
    unsigned sum = 0;
    for(unsigned i = 0; i < n-1; i++)
    {
        sum += input[i];
    }
    return n*(n+1)/2 - sum;
}
```

6. Define a function that returns the average of the minimum and maximum elements in an unsorted array of integers when given the array and the number of elements. How would this code change if the input array is sorted?

   Solution:

```
unsigned average(unsigned* input, unsigned count)
{
    unsigned min = input[0];
    unsigned max = input[0]
```

```
    for(unsigned i = 1; i < count; i++)
    {
        if(input[i] < min) min = input[i];
        if(input[i] > max) max = input[i];
    }
    return (min+max)/2;
}

unsigned averageSorted(unsigned* input, unsigned count)
{
    return (input[0]+input[count-1])/2;
}
```

7. Draw the array represented by `int arr[5];` use null to denote uninitialized memory.

   Solution: (elements separated by commas)

   NULL, NULL, NULL, NULL, NULL

8. Now redraw the array after this code executes:

```
*arr = 1;
*(arr+2) = 5;
```

   Solution:

   1, NULL, 5, NULL, NULL

9. Define a void function that takes a pointer to an integer variable as a parameter, and increments its value by 10. (hint: void functions return type is void)

   Solution:

```
void add10(int* input)
{
    *input += 10;
}
```

10. What is the output of the following code? If it breaks at any point, indicate what went wrong.

```
#include <iostream>

int mystery(int x, int *y) {
```

```
    x = x + 10;
    *y = x * 2;
    return x;
}

int* mystery2() {
    int x = 50;
    return &x;
}

int main() {
    int x = 2, y = 3;
    x = mystery(x, &y);
    std::cout << "(x, y): (" << x << ", " << y << ")" << std::endl;
    int *z = mystery2();
    std::cout << "z: " << *z << std::endl;
}
```

Solution:

x = 12

y = 24

z causes segfault since it is the address of a deallocated local variable

11. Write a program that removes any duplicate integers from an input array and prints the resulting array. The function should also take the number of elements as a parameter. For example, given $[1, 2, 2, 3, 4, 2, 5]$ and 7, the program should print $[1, 2, 3, 4, 5]$.

    Solution:

```
void noRepeats(unsigned* input, unsigned count)
{
    //write stores the index where the next non-repeating element should be written
    unsigned write = 0;
    for(unsigned i = 0; i < count; i++)
    {
        bool repeat = false;
        for(unsigned j = 0; j < i; j++)
        {
            if(input[i] == input[j])
```

```
            {
                write--;
                repeat = true;
                break;
            }
        }
        if(!repeat)
            input[write] = input[i];
        write++;
    }
    std::cout << "[" << input[0];
    for(unsigned i = 1; i < write; i++)
    {
        std::cout << ", " << input[i];
    }
    std::cout << "]" << std::endl;
}
```