Final CS111 Summer 2020

Instructions:
1) Make sure your name and CUNY ID are filled in.

2) When asked to write a program, begin with the main portion of the program. In most cases I have written the beginning of the function for you and you just fill in the function body.

3) The only outside functions you can use are:
a) cout, cin, rand(), srand(), and time()
b) functions you write yourself
c) functions that appear anywhere on this test

SECTION 1 – Programming 5 Points each

Q1) Why is sorting data important. i.e. Sorting an array?
A1) _To find a piece of data._____

Anything wrong with the code below? What if any corrections are needed?
Q2)

```
char* foo1(char* str)
{
        return(str);
}
```

A2) __No, the array memory was declared outside this function.
Q3)

```
char* foo2()
{
        char str[100] = "ABC";
        return(str);
}
```

A3) __Yes, the array disappears when this function finishes.

Q4)

```
int StringLength(const char str[])
{
        int i = 1;
        while (str[i])
                i++;
        return(i);
}
```

A4) __Yes, i should start at position 0.

Q1) Assuming the following array: 1 2 3 4 5 6 7 8 9 10 What order of numbers would cause BubbleSort be slowest?
A) __10 9 8 7 6 5 4 3 2 1

Anything wrong with the code below? What if any corrections are needed?
Q2)

```
char str[100] = "ABC";
char* foo3()
{
        return(str);
}
```

A2) __No, the array returned is global memory.

Q3)

```
void Swap1(char c1, char c2)
{
        char temp = c1;
        c1 = c2;
        c2 = temp;
}
```

A3) _Yes the c1 and c2 need to passed by reference.

Q4)

```
int StringLength(const char str[])
{
        if (!str[0])
                return(0);
        int i = 1;
        while (str[i])
                i++;
        int size = 0;
        while (i>=0)
        {
                size++;
                i--;
        }
        return(size);
}
```

A4) _Yes while loop should be >0.

Section 2 – Programming
Q1) 10 Pts
WaterNeededForOatmeal - Water needed to cook N servings of oatmeal
to cook Oatmeal 1 cup is needed when there is only one
serving. When cooking two servings an additionl 3/4 cup
is needed. see table below

Number of servings: 1 Amount of water: 1 Cups
Number of servings: 2 Amount of water: 1.75 Cups
Number of servings: 3 Amount of water: 2.75 Cups
Number of servings: 4 Amount of water: 3.5 Cups
Number of servings: 5 Amount of water: 4.5 Cups

```
double WaterNeededForOatmeal(const int serving)
{
	int serv = serving;
	double water = 0;
	while (serv > 0)
	{
		if (serv % 2 == 1)
			water += 1;
		else
			water += .75;
		serv--;
	}
	return(water);
}
```

Q2a) 5 Pts
Absolute - returns the absolute value of an integer
Absolute(1)==>1
Absolute(-1) ==> 1

```
int Absolute(int i)
{
	return(i < 0 ? -i : i);
}
```

Q2b) 5Pts)
SumOfEvenNumbers – Sum of even numbers in array
int nums[5] = { 1,2,3,4,5 };
result: 6 cout << SumOfEvenNumbers(nums, 5) << endl;

```
int SumOfEvenNumbers(const int nums[], const int size)
{
	int sum = 0;
	for (int i = 0; i < size; i++)
		if (nums[i] % 2 == 0)
			sum += nums[i];
	return(sum);
}
```

Q3) **MergeStrings** – Merges two strings two characters at a time into dest. First from str1 then str2. If one string is shorter then the remainder of the larger string is copied into the dest.

MergeStrings("AABBCC","DDEEFFGGHH")➔ "AADDBBEECCFFGGHH
MergeStrings("AABB","DDEEFFGGHH")➔ "AADDBBEEFFGGHH

```
char* MergeStrings(char* dest,
        const char* str1,
        const char* str2)
{
        int destCount = 0,count1=0,count2=0;
        while (str1[count1] || str2[count2])
        {
                for (int i = 0; i < 2&& str1[count1]; i++,count1++, destCount++)
                        dest[destCount] = str1[count1];

                for (int i = 0; i < 2&& str2[count2]; i++,count2++, destCount++)
                        dest[destCount] = str2[count2];
        }
        dest[destCount] = 0;
        return(dest);
}
```

Q4) CountSeries – Returns the size of the series a number in a sorted array is part of. Assuming I have a sorted list and I know where one of the items is located (I have the index) return how many items this item is part of a series with a series is considered where all the adjacent number are
different by 1 using the Absolute function
CountSeries({1,2,3,4,5,7,8,9},8,0)==>5
because 1 is part of the series 1,2,3,4,5
CountSeries({1,2,3,20,5,6,7,8},8,1)==>3
because 1 is part of the series 1,2,3
CountSeries({1,2,3,20,5,6,7,8},8,3)==>1
because 20 is not part of any other series

```
int CountSeries(const int nums[],
                const int size)
{
        int count = 1;
        int left = intitialLoc,right= intitialLoc + 1;
        while (left > 0 && Absolute(nums[left] - nums[left - 1]) == 1)
        {
                count++;
                left--;
        }
        while (right < size && Absolute(nums[right] - nums[right - 1]) == 1)
        {
                count++;
                right++;
        }
```

```
        return(count);
}
```

Q5)
ShiftLeft – Deletes a character by shifting string to the left from specified position.
ShiftLeft("ABCDEF", 2)➔"ABDEF"
ShiftLeft("ABCDEF", 0)➔"BCDEF"

```
char* ShiftLeft(char* str, int startPos)
{
        int i = startPos;
        startPos++;
        while (str[startPos])
        {
                str[i] = str[startPos];
                i++;
                startPos++;
        }
        str[i] = 0;
        return(str);
}
```

Section 2 – Programming
Q1a) 3 Pts
Round - rounds a nummber to the nearest whole number
Round(2.1)==>2
Round(1.5)==>2
Round(1.499)==>1

```
int Round(double d)
{
  return(d + .5);
}
```

Q1b) 7 Pts
Pandemic - if there is a spreadRate of 2.7 (one person can infect 2.7 people) and infection is spread every two weeks to new people from newly infected calculate how many people will be infected after n days
result is rounded to the nearsest whole number time must be greater than or equal to one
Pandemic(15,2.7) ==> 3 after 15 days one person will infect approximately 3 new people
Pandemic(29,2.7) ==> 10 after 29 days one person will infect approximately 10 new people
Pandemic(42,2.7) ==> 30 after 42 days one person will infect approximately 30 new people

```
double Pandemic(int days, double spreadRate)
{
        double newInfections = 1;
        int times = days / 14;
        double total = 0;
```

```
        while (times > 0)
        {
                newInfections *= spreadRate;
                total += newInfections;
                times--;
        }
        return(Round(total));
}
```

Q2a 5Pts)
SumIfGreaterThan - returns the sum of all numbers greater than a specific value
SumIfGreaterThan({ 1.1,1.2,2.1,3.1 }, 4, 2.0) ==> 5.2

```
double SumIfGreaterThan(const double numbers[],
        const int size,
        const double minValue)
{
        double sum = 0;
        for (int i = 0; i < size; i++)
        {
                if (numbers[i] > minValue)
                        sum += numbers[i];
        }
        return(sum);
}
```

Q2b 5Pts)
SumOfEvenNumbers – Sum of even numbers in array
int nums[5] = { 1,2,3,4,5 };
result: 6 cout << SumOfEvenNumbers(nums, 5) << endl;

```
int SumOfEvenNumbers(const int nums[], const int size)
{
        int sum = 0;
        for (int i = 0; i < size; i++)
                if (nums[i] % 2 == 0)
                        sum += nums[i];
        return(sum);
}
```

Q3 10 Pts) **ToDouble -** converts a double in string format to a double
ToDouble("1.23") ==>1.23

```
double ToDouble(const char* str)
{
        double upper = 0,lower=0;
        int i = 0;
        while (str[i] && str[i]!='.')
```

```
        {
                upper = upper * 10 + str[i] - '0';
                i++;
        }

        if (str[i] == '.')
                i++;
        double pow = .1;
        while (str[i])
        {
                lower = lower  + (str[i] - '0')*pow;
                pow = pow / 10;
                i++;
        }
        return(upper + lower);
}
```

Q4)

AddAll - Adds all the doubles in the string with each other. Note every double is separated by a semi-colon.

AddAll("1.245;2.9")==>4.145

```
double AddAll(const char* str)
{
        int i = 0;
        double ret = 0;
        while (str[i])
        {
                char number[100];
                int index = 0;
                while (str[i] && str[i] != ';')
                {
                        number[index] = str[i];
                        i++;
                        index++;
                }
                number[index] = 0;
                ret += ToDouble(number);
                i++;
        }
        return(ret);
}

struct Sale
{
        char Person[100];
        double sale;
```

```
        double tax;
};

struct SaleStatistic
{
        double averageSale;
        double totalSale;
        double totalTax;
};
```

Q5) **CalculateStatistic –** Goes through all sales and returns the average sale, total sale, and total tax in SaleStatistic.

```
SaleStatistic CalculateStatistic(Sale* sales, int size)
{
        SaleStatistic sale;
        for (int i = 0; i < size; i++)
        {
                sale.averageSale += sales[i].sale;
                sale.totalSale += sales[i].sale;
                sale.totalTax += sales[i].tax;
        }
        sale.averageSale /= 10.0;

        return(sale);
}
```