Final CS111 Fall 2019 TEST A

Instructions:

1) Make sure your name and CUNY ID are filled in.
2) For programming questions, in most cases, I have written the signature of a function for you, and you just have to fill in the rest of the function.
3) You can only use functions: a)that are used in this test (as questions) b) ones that you write yourself or c) functions that are defined below.
The only exception is that you can use the cout, cin, rand, and the time functions.

```
void Swap(char &c1, char &c2) { char temp = c1; c1 = c2; c2 = temp; }
bool IsLower(char c) { return(c >= 'a'&&c <= 'z'); }
bool IsUpper(char c) { return(c >= 'A'&&c <= 'Z'); }
char ToLower(const char c) { return(IsUpper(c) ? 'a' + c - 'A' : c); }
char ToUpper(const char c) { return(IsLower(c) ? 'A' + c - 'a' : c); }
bool IsAlpha(char c) { return(IsLower(c) || IsUpper(c)); }
bool IsNumeric(char c) { return(c >= '0'&&c <= '9'); }
bool IsAlphaNumeric(char c) { return(IsAlpha(c) || IsNumeric(c)); }
bool IsSpace(char c) { return(c == ' ' || c == '\t' || c == '\r' || c == '\n'); }
void MakeUpper(char &c) { c = ToUpper(c); }
void MakeLower(char &c) { c = ToLower(c); }
bool IsSame(const char c1, const char c2) { return(ToUpper(c1) == ToUpper(c2)); }
int StringLength(const char *str) { int i = 0; while (str[i])i++; return(i); }
int Count(const char *str, const char c) { int i = 0; int count = 0; while (str[i]) { if (str[i] ==
c)count++; i++; }return(count); }
bool Contains(const char *str, const char c) { return(Count(str, c) > 0); }
char * StringCopy(char *dest, const char *src, const int max = 1000000) { int i = 0; while (src[i] && i <
(max - 1)) { dest[i] = src[i]; i++; }dest[i] = 0; return(dest); }
int IndexInArray(const int *nums, const int size, const int num) { for (int i = 0; i < size; i++)if
(nums[i] == num)return(i); return(-1); }
int Absolute(const int num) { return(num < 0 ? -num : num); }
void Swap(int &i1, int &i2) { int temp = i1; i1 = i2; i2 = temp; }
bool HasDecimal(double d) { int i = d; return(i != d); }
int Max(const int *nums, const int size) { int max = nums[0]; for (int i = 1; i < size; i++)if (max <
nums[i])max = nums[i]; return(max); }
int Min(const int *nums, const int size) { int min = nums[0]; for (int i = 1; i < size; i++)if (min >
nums[i])min = nums[i]; return(min); }
int Sum(const int *nums, const int size) { int sum = nums[0]; for (int i = 1; i < size; i++)sum +=
nums[i]; return(sum); }
double Sum(const double *nums, const int size) { double sum = nums[0]; for (int i = 1; i < size; i++)sum
+= nums[i]; return(sum); }
int Mode(const int *nums, const int size) { int count[101]; for (int i = 1; i < 101; i++)count[i] = 0; for
(int i = 0; i < size; i++)count[nums[i]] ++; int most = -1; for (int i = 1; i < 101; i++) { if (count[i] >
most)most = i; }return(most); }
bool Contains(const int *answer, const int size, const int searchFor) { for (int i = 0; i < size; i++)if
(answer[i] == searchFor)return(true); return(false); }
bool IsSpecialCharacter(char c) { return(c == '!' || c == '@' || c == '#' || c == '~' || c == '%'); }
int * BubbleSort(int *nums, const int size) { bool bSwapped = true; while (bSwapped) { bSwapped = false;
for (int i = 0; i < size - 1; i++)if (nums[i] > nums[i + 1]) { Swap(nums[i], nums[i + 1]); bSwapped =
true; } }return(nums); }
int NumberOfDigits(const int i) { int num = i; int numOfDigits = 0; do { num /= 10; numOfDigits++; } while
(num); return(numOfDigits); }
void PrintArray(const int *arr, const int size) { int bufSize = NumberOfDigits(Max(arr, size)) + 1; for
(int i = 0; i < size; i++) { if (i) { for (int j = 0; j < bufSize - NumberOfDigits(arr[i - 1]); j++)cout
<< " "; }cout << arr[i]; }cout << endl; }
bool StringCompare(const char *str1, const char *str2){int i = 0;while (str1[i] && str1[i] ==
str2[i])i++;return(str1[i] == str2[i]);}
```

SECTION 1 – Miscellaneous Questions 5 Points Each

Q1 – What is the output of f1()?

Q1) __2_____

```
void f1()
{
        char str[10] = "ABCDEFGH";
        str[5 / 2] = 0;
        cout << StringLength(str) << endl;
}
```

Q2 – What is the output of f2()?

Q2) A C E G I _____

```
void f2()
{
        for (int i = 0; i < 10; i += 2)
        {
                char c = 'A' + i;
                cout << c << " ";
        }
        cout << endl;
}
```

Q3 – What is the output of f3()?

Q3) _12_____

```
void f3()
{
	int num = 12345;
	for (int i = 0; i < 3; i++)
	{
		num /= 10;
	}
	cout << num << endl;
}
```

Q4 – What is the output of f4()?

Q4) _ A D G__

```
void f4()
{
	char arr[3][3] = {
		{'A','B','C'},
		{'D','E','F'},
		{'G','H','I'} };
	for (int i = 0; i < 3; i++)
		for (int j = 0; j <1; j++)
			cout << arr[i][j] << " ";
	cout << endl;
}
```

SECTION 2 – Basic Programming – Program the three function below. 10 Points Each

Q1 - RandomDigit – Returns a random digit, e.g. a number from 0 through 9.
Sample Output:
RandomDigit() ➔ 0
RandomDigit() ➔ 9

```
// whole number between 0 and 9
int RandomDigit()
{
        srand(time(0));
        return(rand() % 10);
}
```

Q2 - Statistics - Writes the smallest, largest, average and sum.
Sample Output :
int nums[10] = { -1,2,3,4,5,6,7,8,9,10 };
Statistics(nums, 10) ==> min = -1 max = 10 average = 5.3 sum = 53

```
void Statistics(int *nums, const int size)
{
        int min = nums[0], max = nums[0];
        double sum = 0.0;

        for (int i = 0; i < size; i++)
        {
                if (min > nums[i])
                        min = nums[i];
                if (max < nums[i])
                        max = nums[i];
                sum += nums[i];
        }
        cout << "Min = " << min <<
                " Max = " << max <<
                " Sum = " << sum <<
                " Average = " << sum / size;
}
```

Q3 PrintStrings – Prints out the strings of the array in one line

Sample Output:

```
char strings[3][10] = { {"Hi "},
                        {"everyone"},
                        {" !"} };
```

PrintStrings(strings) ➔ "Hi everyone !"

```
void PrintStrings(char strings[3][10])
{
        for (int i = 0; i < 3; i++)
                cout << strings[i];
        cout << endl;
}
```

SECTION 3 –Programming – Program the following functions 10 Points Each Choose 4 from the six below.

Q1) Subset - Returns true if all members of subset are also members of set
Suggestion - Use the Contains function
Sample Output:
int nums[10] = { 1,2,3,4,5,6,7,8,9,10 };
int nums2[5] = { 3,4,5,11,12};
int nums3[3] = { 3,4,5};
bool Subset(nums, 10,nums2,5) ➔ false
bool Subset(nums, 10,nums3,3) ➔ true

```
bool Subset(const int *set, const int setSize,
        const int *subset, const int subsetSize)
{
        for (int i = 0; i < subsetSize; i++)
                if(!Contains(set,setSize,subset[i]))
                        return(false);
        return(true);
}
```

Q2 – Intersect -This function outputs the characters that two arrays have in common.
Sample Output:
int nums[10] = { 1,2,3,4,5,6,7,8,9,10 };
int nums2[5] = { 3,4,5,11,12 };
int nums3[3] = { 10,11,12 };
Intersect(nums,10,nums2,5)➔ 3 4 5
Intersect(nums,10,nums3,3)➔ 10

```
void Intersect(const int *i1, const int iSize1,
        const int *i2, const int iSize2)
{
        for (int i = 0; i < iSize1; i++)
                if (Contains(i2, iSize2, i1[i]))
                        cout << i1[i] << " ";
        cout << endl;
}
```

Q3- Duplicates – Writes all the duplicates in same array

Sample Output:
char letters[10] = "ABCDAFGGH";
char letters1[6] = "ABCDE";
char letters2[4] = "AAA";
Duplicates(letters➔A G
Duplicates(letters1)➔
Duplicates(letters2)➔ A A A

```
void Duplicates(const char *letters)
{
	int i = 0;
	while (letters[i])
	{
		int j = i + 1;
		while (letters[j])
		{
			if (letters[i] == letters[j])
				cout << letters[i] << " ";
			j++;
		}
		i++;
	}
	cout << endl;
}
```

Q4 – ValidatePassword – Validates a password to make sure it meets certain criteria. If all criteria are met, **validPassord** is returned, otherwise one of the other **PasswordError** values will be returned.
a) everything ok, returns **validPassword**
b) at least 10 characters, < 10 characters returns **tooShort**
c) at least one upper case character, no uppercase returns **noUpperCase**
d) at least one lower case, no lowercase returns **noLowerCase**
e) at least one special character (not A- Z or 0 - 9), no special character returns **noSpecialCase**
f) no spaces, spaces returns **containsSpaces**

enum PasswordError{validPassord, tooShort, noUpperCase, noLowerCase, noSpecialChar, containsSpaces};

Sample Output:
ValidatePassword("a") ➔ tooShort
ValidatePassword("aaaaaaaaaaaaa")➔ noUpperCase
ValidatePassword("Aaaaaaaaaaa")➔ noSpecialChar
ValidatePassword("Aa %aaaaaaaaaa")➔ containsSpaces
ValidatePassword("Aa%aaaaaaaaaa")➔ validPassword

```
PasswordError ValidatePassword(const char *pswd)
{
	if (StringLength(pswd) < 10)
		return(tooShort);

	if (Contains(pswd, ' '))
		return(containsSpaces);

	bool specialChar = false, upper = false, lower = false;
	int i = 0;

	while (pswd[i])
	{
		if (IsUpper(pswd[i]))
			upper = true;

		if (IsLower(pswd[i]))
			lower = true;

		if (!IsAlphaNumeric(pswd[i]))
			specialChar = true;
		i++;
	}
	return(specialChar == false ? noSpecialChar :
		(upper == false ? noUpperCase :
		(lower == false ? noLowerCase : validPassord)));
}
```

Q5 -ReverseOfEachOther – Compares two strings to see if they are the reverse of each other.
Sample Output:
ReverseOfEachOther("","") ➔ true
ReverseOfEachOther("ABC","CBA") ➔ true
ReverseOfEachOther("ABCDE","FECBA") ➔ false
ReverseOfEachOther("ABCDE","DCBA") ==> false

```
bool ReverseOfEachOther(const char *str1, const char *str2)
{
	if (StringLength(str1) != StringLength(str2))
		return(false);

	for (int i = 0,j= StringLength(str2)-1; i < StringLength(str1); j--,i++)
	{
		if (str1[i] != str2[j])
			return(false);
	}
	return(true);
}
```

Q6 - RandomString – Returns a random string of characters 0..1 a..z and A..Z. The returned string has **size** random characters.
Sample Output:
RandomString(str, 5) ➔"s2aIO"
RandomString(str, 1) ➔"h"
RandomString(str, 10) ➔"oLe9Z3QXoS"

```
char *RandomString(char *str,int size)
{
	srand(time(0));

	for (int i = 0; i < size; i++)
	{
		int choice = rand() % 3;
		str[i] = choice == 0 ? '0' + rand() % 10 :
			(choice == 1 ? 'A' + rand() % 26 :
				'a' + rand() % 26);
	}
	str[size] = 0;
	return(str);
}
```