MIDTERM 3 CS111 Fall 2019 ANSWERS

SECTION 1 – Miscellaneous Questions 5 Points Each

Q1 – What is the output of f1()?
Q1 - Draw answer below…

A E
B D
C
B D
A E

```
void f1()
{
        char abc[5][6] = {  "ABCDE",
                            "ABCDE",
                            "ABCDE",
                            "ABCDE",
                            "ABCDE" };
        for (int i = 0, end = 4; i < 5; end--, i++)
        {
                for (int j = 0; j < 5; j++)
                {
                        if (i == j || end == j)
                                cout << abc[i][j];
                        else
                                cout << " ";
                }
                cout << endl;
        }
}
```

Q2 – What is the output of Order() function below for the following input?
Order('a', 'A'); ➔ a A
Order('A', 'a'); ➔ a A A a
Order('Z', 'a'); ➔a Z Z a
Order('a', 'z'); ➔ a z

```
void Order(const char c1, const char c2)
{
	if (ToUpper(c1) == ToUpper(c2))
	{
		if (IsUpper(c1) && IsLower(c2))
			cout << c2 << " " << c1 << endl;
	}
	else
	{
		if (ToUpper(c1) > ToUpper(c2))
			cout << c2 << " " << c1 << endl;
	}
	cout << c1 << " " << c2 << endl;
}
```

Q3 – a) Which input will make the Max function below to crash?
EMPTY ARRAY

```
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);
}
```

Q3 - b) Which input will make the Max1 function below to give a wrong answer?

Array of all negative numbers

```
int Max1(const int *nums, const int size)
{
	int max = 0;
	for (int i = 0; i < size; i++)
		if (max < nums[i])
			max = nums[i];
	return(max);
}
```

Q4) – StringCopy – re-write the StringCopy function below with a for loop instead of a while loop.

```
char * StringCopy(char *dest, const char *src)
{
        int i = 0;
        while (src[i])
        {
                dest[i] = src[i];
                i++;
        }
        dest[i] = 0;
        return(dest);
}
```

Q4) Answer

```
char * StringCopy(char *dest, const char *src)
{
        for (int i = 0; src[i]; i++)
        {
                dest[i] = src[i];
                dest[i+1] = 0;
        }
        return(dest);
}
```

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

Q1 - StringNCompare - Compares two strings for the first max characters. If those characters are the same then return true, otherwise false.
Sample Output:
StringNCompare("FRED1", "FRED2") ➔ false
StringNCompare("FRED1", "FRED2",4) ➔ true
StringNCompare("FRED1", "FRED2",0) ➔ true
StringNCompare("FBED2", "FRED2",2) ➔ false

```
bool StringNCompare(const char *str1, const char *str2, int num = 1000000)
{
	if (num == 0)
		return(true);

	int i = 0;
	int charsCompared = 0;
	while (str1[i]&& str1[i] == str2[i])
	{
		charsCompared++;
		if (charsCompared == num)
			break;
		i++;
	}
	return(str1[i] == str2[i]);
}
```

Q2 - IsEven - Returns true or false depending on whether the input is even.
Write this function using the conditional operator, also known as the ? (question operator).
Sample Output :
IsEven(-2) ➔ true
IsEven(3) ➔ false

```
bool IsEven(int i)
{
	return(i % 2 == 0 ? true : false);
}
```

Q3 - SumEvenNumbers - Returns sum of all numbers of the array that are even.
Sample Output :
int nums[10] = { 1,2,3,4,5,6,7,8,9,10 };
SumEvenNumbers(nums,10) ➔ 30
int nums1[10] = { 1,3,5,7,9,11,13,15,7,0 };
SumEvenNumbers(nums1, 10) ➔ 0

```
int SumEvenNumbers(const int *nums, int size)
{
	int sum = 0;
	for(int i = 0; i < size; i++)
		if(IsEven(nums[i]))
			sum += nums[i];
	return(sum);
}
```

SECTION 3 –Programming – Program the following functions – Questions 1 - 5 10 Points Each, Question 6 20 Points

Q1 - CompareEndOfStrings - compares 2 strings starting from the end for numToCompare amount of characters

Sample Output:

CompareEndOfStrings("FRED", "BED", 2) ➔ true
CompareEndOfStrings("FRED", "BED", 3) ➔ false
CompareEndOfStrings("FRED", "BED", 0) ➔ true
CompareEndOfStrings("FRED", "FRED", 10) ➔ true

```
bool CompareEndOfStrings(const char *str1,
        const char *str2,
        int numToCompare)
{
        if (numToCompare <= 0)
                return(true);

        int i = StringLength(str1)-1;
        int j = StringLength(str2)-1;
        int charsCompared = 0;

        while (i>=0&& j >=0)
        {
                if (str1[i] != str2[j])
                        return(false);

                charsCompared++;

                if(charsCompared == numToCompare)
                        return(true);

                i--;
                j--;
        }
        return(true);
}
```

Q2 – Scramble -This function will randomly scramble each letter of the string with another letter at least once.
Suggestion - Use the StringLength and the Swap function
Sample Output:
Scramble("ABCD") ==> "BADC"
Scramble("ABCDE") ==> "EABCD"

```
char * Scramble(char *text)
{
	srand(time(0));
	int i = 0;
	while (text[i])
	{
		Swap(text[i], text[rand() % StringLength(text)]);
		i++;
	}
	return(text);
}
```

Q3 - CreatePalindrome – takes any string that is passed in and turns it into a Palindrome
Sample Output:
CreatePalindrome("ABA") ==> "ABAABA"
CreatePalindrome("ABC") ==> "ABCCBA"
CreatePalindrome("") ==> ""

```
char * CreatePalindrome(char *str)
{
	int i = StringLength(str)-1;
	int j = StringLength(str);
	while (i >= 0)
	{
		str[j] = str[i];
		i--;
		j++;
	}
	str[j] = 0;
	return(str);
}
```

Q4 - BubbleSortIgnoreCase - sort the string using the Bubble Sort algorithm alphabetically ignoring character case, i.e. upper and lower case of the same character are identical.
Sample Output:

char str1[100] = "cddchhhaabbb";
BubbleSortIgnoreCase(str1) ➔ aabbbccddhhh

char str2[100] = "cDdChhHAaBbb";
BubbleSortIgnoreCase (str2) ➔ AaBbbcCDdhh

```
char * BubbleSortIgnoreCase(char *str)
{
	bool swapped = true;
	while (swapped)
	{
		swapped = false;
		for (int i = 0; i < StringLength(str) - 1; i++)
			if (ToUpper(str[i]) > ToUpper(str[i + 1]))
			{
				Swap(str[i], str[i + 1]);
				swapped = true;
			}
	}
	return(str);
}
```

Q5) RandomDecimal – return a random decimal number that has up to three digits after the decimal.

Sample Output:
RandomDecimal(1, 2) ➔ 1.682
RandomDecimal(20, 35) ➔ 27.981
RandomDecimal(20, 35) ➔ 28.972
RandomDecimal(20, 35) ➔ 28.676

```
double RandomDecimal(const int low, const int high)
{
	srand(time(0));
	int l = low * 1000;
	int h = high * 1000;
	return((l + (rand()*1000) % (h - l + 1))/1000.0);
}
```

Q6) **LetterCount** - Show the letter that appears most often. If multiple characters appear the same amount of times, return the first character.

a) note there are 26 letters in the alphabet
b) the function is case insensitive, 'A' and 'a' are treated equally
c) non alpha characters are ignored.
d) Suggested helper functions: `IsAlpha, Max, IndexInArray, ToUpper`

Sample Output:
LetterCount("&&&")➔ #
LetterCount("aaBBBccccDDDDD") ➔ D
LetterCount("aaABBBcccDDD") ➔ A

```
char LetterCount(const char *str)
{
	int letters[26];
	for (int i = 0; i < 26; i++)
		letters[i] = 0;
	int i = 0;
	while (str[i])
	{
		if (IsAlpha(str[i]))
			letters[ToUpper(str[i]) - 'A']++;
		i++;
	}
	if (Max(letters, 26) == 0)
		return('#');

	return('A' + IndexInArray(letters, 26, Max(letters, 26)));
}
```