Programming Assignment 3

You can choose one of two options for this assignment:

1. **Write a program to practice working with arrays of chars terminated using special char '\0'.**

Requirement:

- Write a function that tests whether a given char array stores a palindrome, i.e., whether the word/sentence stored in the array reads the same forward or backward. E.g., **madam** is a palindrome, while **world** is not.

```
// str is an array of char, terminated using '\0'
// return true if str stores a string of char that is palindrome; false otherwise
bool IsPalindrome (char str[ ])
// Hint: compare first char with last char, second char with second last char, …
```

- Write a function that tests whether a given char array has duplicate chars. For example, **madam** contains duplicates (two m's), while **world** does not contain duplicates.

```
// test whether an array of char, str, contains duplicates
// the array str stores chars, terminated using '\0'
bool containDuplicates (char str[ ])
```

- Write a function that deletes a char from the array and move the chars after it forward so that there is no "hole" in the array.

```
// Remove char at given index from the array str, and move all
// chars after the index up by one slot
// e.g., if str's contains 'a', 'b', 'c','d','\0', and index is 1
 //  after the function, str contains 'a','c','d','\0'
  void removeChar (char str[ ], int index)
```

- Write your main to test these functions one by one.

2. Write a program to perform addition of two large positive integers which can have with up to 10 decimal digits. If the result of the addition is more than 10 digits long then simply give the output message "Addition overflow".

Please feel free to base your program on the code given in the end of the document. While on storm, you can use the following command to copy it to you directory.

$~zhang/bin/copyLab3

Here is the link to the code:

http://storm.cis.fordham.edu/zhang/cs2000/Labs/lab3.cpp

Requirement:

- Use **int** arrays to store decimal numerals. As this is sometimes partially filled, we can use a separate int (e.g., decNumberLen) to store the number of digits.

```
const int MAX_CAP=10;
int decNumberDigits[MAX_CAP];
int decNumberLen=0;
    // decNumberDigits[0] stores the 1-th digit
   // decNumberDigits[1] stores the 10-th digit
    // decNumberDigits[2] stores the 100-th digit
```

for example, the number 9845 will be represented as the following

```
int num1[MAX_CAP]={5,4,8,9};
int num1_length=4;
```

- Design and implement a function that displays such large integer in standard output.
- Design and implement a function that performs addition on two large integers.
- Function ReadLargeInt() has been provided. It reads such large integer from standard input.
- Test above functions one by one, and then write a main that works as follows:

Repeatedly ask the user to enter two large integers, display back the two integers, and then display the sum of the two numbers. Ask the user whether to continue.

- To submit: submit2000 LAB3 lab3.cpp

```
#include <iostream>
#include <stdio.h>
using namespace std;

const int MAX_CAP=10;

/* Display an big integers stored as an array of decimal digits
   bigInt[0] is 1-th digit
   bigInt[1] is 10-th digit
    ...
   the number of digits is given by len
*/
void Display(int bigInt[], int len)
{
   //todo by you. You need to display the most-significant digit first, and 1-th digit last

}

/* Add two large integers and store the result in res
  aDigits and aLen give the first integer, bDigits and bLen give the second integer
  sumDigits and sumLen will store the sum
  return true fi successful; false if fail (due to overflow)
  */
bool AddLargeInt (int aDigits[], int aLen, int bDigits[], int bLen,
        int sumDigits[], int & sumLen)
{
        //Todo by you
        //1. Pad the two operands with leading 0's, i.e., add 0's to the end of
        // aDigits, bDigits array. Recall the capacity of both arrays is MAX_CAP



        int carry=0;
        //2. adding the two large integers, below is the pseudocode
        // for i=0 to MAX_CAP-1  //start from 1-th digit, 10-th digit, ...
        //      add the aDigits[i], bDigits[i] and carry up, and set to sum
        //      set carry to 1 if the above sum is >=10
```

```
	//	set sum[i] to sum-carry

	//if carry!=0
	//   overflow, return false
	//otherwise
	//   return true
	return true; //for now

}

/* Read large integers of up to MAX digits, store the digits in the array, and set the
   digit len
   return true if the input is valid, false if otherwise
   */
bool ReadLargeInt (int digits[], int & digit_len)
{
	string input;
	int dig_value;

	cout <<"Enter large int (up to "<<MAX_CAP <<"  digits):";
	cin >> input;
	//cout <<"You enter:"<<input<<endl;

	digit_len = input.length(); //call the method on string class to get the length of input
	if (digit_len>100)
	if (digit_len>100)
	{
		cout <<"The input is too long\n";
		return false;
	}

	for (int i=0;i<digit_len;i++)
	{
		dig_value = input[digit_len-i-1]-'0'; //convert char to int value
			//scan input string from the end to the front
			// when i=0, input[digit_len-1] is accessed, converted to 0,1..9
			// when i=1, input[digit_len-2] ....

		if (dig_value>9 || dig_value<0)
		{
			cout <<"Invalid char " << input[digit_len-i-1]<< " in the input"<<endl;
			return false;
		}
		digits[i]=dig_value;
	}
	return true;
```

```
}

int main()
{
   int num[MAX_CAP]={0,0,0,0,0,5,4,8,9};
   int num_len=9;

   int num2[MAX_CAP]={1,0,0,5,0,1};
   int num_len2=6;

   //Test Display() function
   Display(num,num_len);
   cout <<"+";
   Display(num2,num_len2);

   //Test AddBigInt ...


   //Test ReadLargeInt ...
   bool ret;
   do{
       ret = ReadLargeInt (num, num_len);
   } while (!ret);

   cout <<"Read this:";
   Display (num, num_len);


   //Todo:
   //Main loop that allows user to enter two large ints, display the sum

}
```