Midterm Assignment Section B

In this assignment, your task is to write a class in C++ for Bitwise operations of numbers.

Implementation:

Class Name: BitSet

Possible Variables:

int *bitArray //holds the bit values

A **BitSet** class creates a special type of array that holds bit values (0/1). The array size is 16 by default. The **BitSet** constructors are shown here:

BitSet( )

BitSet(int *size*)

The first version creates a **BitSet** object with *default size*. The second version allows you to specify its initial size (that is, the number of bits that it can hold). In both case, **all bits are initialized to zero**.

In the **BitSet the index of **LSB** and **MSB** are 0 and *size-1* respectively.

Methods	**Description**
int cardinality( )	Returns the number of set bits in the invoking object.
void clear( )	Zeros all bits.
void clear(int *index*)	Zeros the bit specified by *index*.
void set(int *index*)	Sets the bit specified by *index*.
void set(int *startIndex*, int *endIndex*)	Sets the bits from *startIndex* to *endIndex*–1.

BitSet get(int *startIndex*,int *endIndex*)	Returns a **BitSet** that consists of the bits from *startIndex* to *endIndex*–1. The invoking object is not changed.
void and(BitSet *bitSet*)	ANDs the contents of the invoking **BitSet** object with that specified by ***bitSet***. **The result is placed into the invoking object.**
void or(BitSet *bitSet*)	ORs the contents of the invoking **BitSet** object with that specified by ***bitSet***. **The result is placed into the invoking object.**
void xor(BitSet *bitSet*)	XORs the contents of the invoking **BitSet** object with that specified by ***bitSet***. **The result is placed into the invoking object.**
boolean intersects(BitSet *bitSet*)	Returns **true** if at least one pair of corresponding bits within the invoking object and ***bitSet*** are 1.
void andNot(BitSet *bitSet*)	***For each 1 bit in ***bitSet***, the corresponding bit in the invoking **BitSet** is cleared.
void bitReplaceLeft(BitSet bitSet, int p, int n)	***Starting from **p-th** bit of the invoking object, replace its **n-bits including the p-th bit**, by the **Leftmost n** bits of ***bitSet***.
void bitReplaceRight(BitSet bitSet, int p, int n)	***Starting from **p-th** bit of the invoking object, **replace n-bits including the p-th bit**, by the **Rightmost n** bits of ***bitSet***.
void show()	Prints the BitSet

*** scroll down for demonstration of the three functions (**page 4**)

****** For Bitwise operations, size of both **BitSet**s must be same. If not, then sign extend the smaller **BitSet**. *(If MSB=0 extend by 0, else extend by 1)*

e.g. If bs1=10001010 and bs2= 10101, then after sign extension bs2=**111**10101

```
int main(){
            BitSet bs1(8), bs2(8);
/*write code for setting the odd bits of bs1*/
/*write code for setting the even bits of bs2*/
            cout<<bs1.cardinality()<<endl;
            cout<<bs2.cardinality()<<endl;
            BitSet tempBS = bs2.get(2,7);
            tempBS.show();
            tempBS.and(bs1);
             tempBS.show();
            tempBS.or(bs2);
            tempBS.show();
            tempBS.xor(bs1);
            tempBS.show();
            if(bs2.intersects(tempBS))
                        cout<< “Intersection!!!”<<endl;
            else
                        cout<< “No Intersection!!!”<<endl;
            tempBS.andNot(bs2);
            tempBS.show();
            bs1.bitReplaceLeft(bs2, 5, 4);
            bs1.show();
            bs2.bitReplaceRight(tempBS, 5, 4);
            bs2.show();
}
```

BitSet:	1	0	0	1
Bit position:	3rd bit	2nd bit	1st bit	0th bit

<u>void andNot(BitSet *bitSet*):</u>

Let, bs1= 1001 and bs2=1010

Then bs1.andNot(bs2) will clear the 1st and 3rd bits of bs1. (*indexing starts with 0)*

So, bs1= **0**0**0**1

<u>void bitReplaceLeft(BitSet bitSet, int p, int n):</u>

Let, bs1= 1**00**1 and bs2=**10**10

bs1.bitReplaceLeft(bs2 , 2, 2) will replace two bits of **bs1** by **leftmost** two bits (10) of **bs2**.

Which two bits of bs1 will be replaced?

Starting from 2nd bit, replace 2 bits including 2nd bit; i.e. the 2nd and 1st bits (00).

So after the operation bs1 will be: 1**10**1

<u>void bitReplaceRight(BitSet bitSet, int p, int n)</u>

Let, bs1= 1**00**1 and bs2=10**10**

bs1.bitReplaceRight(bs2 , 2, 2) will replace two bits of **bs1** by **rightmost** two bits (10) of **bs2**.

Which two bits of bs1 will be replaced?

Starting from 2nd bit, replace 2 bits including 2nd bit; i.e. the 2nd and 1st bits (00).

So after the operation **bs1** will be: 1**10**1

Submission Date:	**B1: 23/06/2012 (Saturday)**	**B2: 25/06/2012 (Monday)**