National University of Computer and Emerging Sciences, Lahore Campus

Course:	**Computer Programming**	**Course Code:**	**CS-103**
Program:	**BS (Computer Science)**	**Semester:**	**Spring 2018**
Duration:	**60 Minutes**	**Total Marks:**	**30**
Paper Date:	**12-Apr-18**	**Weight**	**15 %**
Section:	**All**	**Page(s):**	**1**
Exam:	**Midterm-II**	**Reg. No.**	

Question [30 Marks] Complete the definition of the class given below such that the main program runs successfully. Make sure that your program doesn't consume extra memory space and it should not leak any memory.

```
class BinaryNum
{
private:
      int* binNum;                    //integer array to save binary number
      int noOfBits;                   //total no. of bits
public:
      void Print()
      {
            if(binNum != 0)
            {
                  for(int i = 0 ; i< noOfBits ; i++)
                        cout<<binNum[i];
            }
            cout<<endl;
      }
};
void main()
{
      BinaryNum b1;                         //noOfBits = 0, binNum is NULL
      BinaryNum b2("101");                  //noOfBits = 3, binNum is {1,0,1}
      BinaryNum b3("1001");                 //noOfBits = 4, binNum is {1,0,0,1}

      cout<<"b1 = ";b1.Print();             //Prints Nothing
      cout<<"b2 = ";b2.Print();             //Prints 101
      cout<<"b3 = ";b3.Print();             //Prints 1001

      b1 = b2+b3;
      cout<<"b1 = ";b1.Print();             //Prints 1110
      cout<<"b1[0] = "<<b1[0]<<endl;        //Prints 1 (0th bit in b1)
      cout<<"b1[3] = "<<b1[3]<<endl;        //Prints 0 (3rd bit in b1)

      (b3++).Print();                       //Prints 1001
      b3.Print();                           //Prints 1010
      b1 = "111" + b2;
      b1.Print();                           //Prints 1100
}
```

Solution:

```
class BinaryNumber {
      friend BinaryNumber operator+(char*,BinaryNumber&);

private:
      int* bits;
      int size;

      void createTemporaryOperandsForAddition(int&rs,int*& ra,int*& rb, int
thissize,int othersize,int *thisbits, int *otherbits);

public:
      BinaryNumber();
      BinaryNumber(char* num);
      BinaryNumber(const BinaryNumber&);
      BinaryNumber operator+(const BinaryNumber&);
      BinaryNumber operator++(int);
      BinaryNumber& operator=(const BinaryNumber&);
      int operator[](int);
      void print();
      ~BinaryNumber();
};

BinaryNumber::BinaryNumber(){
      size = 0;
      bits = NULL;
}

BinaryNumber::BinaryNumber(char* num){
      size = strlen(num);
      bits = new int[size];
      for(int i=0; i < size; i++){
            if (num[i] >= '0' && num[i] <= '9')
                  bits[i] = num[i] - '0';
            else {
                  cout << "Not a valid binary number";
                  delete [] bits;
                  size = 0;
                  bits = NULL;
                  break;
            }
      }
}

BinaryNumber::BinaryNumber(const BinaryNumber& num){
      this->size = num.size;
      this->bits = new int[size];
      for(int i=0; i < size; i++){
            this->bits[i] = num.bits[i];
      }
}

BinaryNumber& BinaryNumber::operator=(const BinaryNumber& num){
      if (this->bits != 0)
            delete [] this->bits;

      this->size = num.size;
      this->bits = new int[size];
      for(int i=0; i < size; i++){
            this->bits[i] = num.bits[i];
      }
}

int BinaryNumber::operator[](int index){
      if(index > 0 && index < size)
            return bits[index];
      else {
            cout << "Out of bounds" << endl;
      }

      return 0;
}

BinaryNumber BinaryNumber::operator++(int){
      BinaryNumber temp = *this;
      (*this) = (temp + BinaryNumber("1"));
      return temp;
}

void BinaryNumber::createTemporaryOperandsForAddition(int&rs, int*& ra, int*&
rb, int thissize,int othersize,int *thisbits, int *otherbits){
            ra = thisbits;
            rs = thissize;
            rb = new int[rs];
            int diff = thissize-othersize;
            for(int i=0; i < diff; i++)
                  rb[i] = 0;
            for(int i = 0; i < othersize; i++)
                  rb[i + diff] = otherbits[i];
}

BinaryNumber BinaryNumber::operator+(const BinaryNumber& num){
      int rs;
      int *rt, *ra, *rb;

      // create same size temporary arrays and fill with leading zeros
      if(this->size > num.size){
            createTemporaryOperandsForAddition(rs,ra,rb,
                                    this->size,num.size,this->bits,num.bits);
      }
      else{
            createTemporaryOperandsForAddition(rs,ra,rb,
                                          num.size,this->size,num.bits,this->bits);
      }

      // add
      int carry = 0;
      rt = new int[rs];
      for(int i=rs-1; i >= 0; i--){
            rt[i] = ra[i] + rb[i] + carry;
            if(rt[i] >= 2){
                  rt[i] %= 2;
                  carry = 1;
            }
            else{
                  carry = 0;
            }
      }

      // free memory for temporary array
      delete [] rb;

      // produce final result

      BinaryNumber val;

      if(carry == 1){
            val.size = rs + 1;
            val.bits = new int[val.size];
            val.bits[0] = carry;
            for(int i=0; i < rs; i++){
                  val.bits[i+1] = rt[i];
            }
            delete [] rt;
      }
      else{
            val.size = rs;
            val.bits = rt;
      }

      return val;
}

BinaryNumber operator+(char* a, BinaryNumber& b){
      BinaryNumber t(a);
      BinaryNumber c = (t + b);
      return c;
}

void BinaryNumber::print(){
      for(int i=0; i < size; i++){
            cout << bits[i];
      }
      cout << endl;
}

BinaryNumber::~BinaryNumber(){
      if(bits != NULL){
            delete [] bits;
      }
}
```