ISTANBUL KULTUR UNIVERSITY
COMPUTER ENGINEERING
OBJECT ORIENTED PROGRAMMING, HOMEWORK #2

According to following program execution, implement the **inf_int** class according to related *constructors, member functions and operator overloadings.*

```
int main() {
//constructors
inf_int x1(19);          //parameter is int therefore you need to convert
inf_int x2(“125”);
inf_int x3;              //value is “0”, default constructor
inf_int x4(-97);         //be careful about minus sign
inf_int x5(“-35”);       //be careful about minus sign

//member functions
x2.deleteFirstDigit();           //new value is “25”
x3.setValue(42)                  //new value is “42”
x1.add(x2);                      // new value is ”44”
x1.changeSign();                 // new value is ”44” with the related sign
x1.print();                      // write “-44” to the screen
x3.concatenate(x2);              //new value is ”4225”, string addition
x4=x3;                           // deep copy for dynamic variables

//operator overloading
x3++;                            // new value is “4226”
                                 //operator overloading for post increment
                                 // BE CAREFUL ABOUT MINUS SIGN
x5=x3+x2;                        // new value is “4129”
                                 //operator overloading for addition
                                 // BE CAREFUL ABOUT MINUS SIGN
}
```

Additional Imlementations

1) Write a copy constructor as follows;

```
        inf_int(const inf_int& x);          // (1) copy constructor
                                            // other member functions should be here but they are not shown.
```

2) Write a destructor for the class. (be careful about copying the dynamic variable(s) in the implementation of member functions)

Notes:

- The class must start as follows.
- You can write additional instance variables and functions.
- However you need to use digit as (char*) for storing data and processing it.
- **BE CAREFUL THERE IS NO SIZE LIMIT FOR INF_INT.**

```
class inf_int{                              // infinite integer class
private:
        char* digit;                        // points to a string of digits. Perform dynamic allocation when necessary.
        unsigned int length;                // stores the number of actual digits
        bool thesign;                       // we assume thesign is 1 if positive integer, and 0 if negative integer.
        //********
public :
        //********

};
```