National University of Computer and Emerging Sciences, Lahore Campus

Course:	**Computer Programming**	**Course Code:**	**CS103**
Program:	**BS(Computer Science)**	**Semester:**	**Fall 2017**
Duration:	**180 Minutes**	**Total Marks:**	**70**
Paper Date:	**14-Dec-2017**	**Weight**	**40%**
Section:	**All**	**Page(s):**	**10**
Exam:	**Final**	**Roll No:**	
Name:		**Section:**	

Instruction/Notes: -You can take extra sheets for rough work but do not attach with this paper.
-In case of confusion or ambiguity make a reasonable assumption. Questions are not allowed

Question # 1: [Marks: 5*3 = 15]

Write the output of code segments given below and if there is any error, mention clearly.

Part (A)

```
class Number{
public:
	int* value;
	Number(int v) {
		value = new int(v);
		cout << "Value: " << *value << endl;
	}
	~Number() {
		cout << "Killed: " << *value << endl;
		delete value;
	}
};
class Question{
public:
	Number marks;
	Question(int A) : marks(A) {
		cout << "New Object \n";
	}
	Question(const Question &X) : marks(*X.marks.value + 10){
		cout << "ItsEasy" << endl;
	}
};
void Difficult(Question why){
	Question Quest = why;
}
void main(){
	Question Answer(1);
	Difficult(Answer);
}
```

Output/Error:

Part (b)

```
class Yes{
public:
Yes() { cout <<" Yes() "; }
};

Class No{
 Yes y;
public:
No() { cout <<" No() "; }

};

Class Emotion{
public:
Emotion(){
cout <<"Emotion() "; }
};
```

```
Class Sad : public Emotion{
No n;
public:
 Sad(){ cout <<"Sad() "; }
};

Class Depress : public Sad{
public:
 Depress(){cout <<" Depress() "; }
};

void main(){
      Depress why;
      cout <<"\n OH No! :( \n";
      Sad noWay;
}
```

Output/Error:

Part (c)

```
Class Parent{
      int* b;
public:
Parent(){ b = new int(10); }
Virtual void Print(){
cout <<"B = "<< *b<< endl;}
~Parent(){ delete b; }
};
Class Child : public Parent{
int* d;
public:
Child(){ d = new int(20); }
void Print(){
Parent::Print();
cout <<"D = "<< *d << endl;
}
~Child(){ delete d; }
};
```

```
void main()
{
      Parent* pPtr = new Child();
      pPtr->Print();
      delete pPtr;
}
```

Output/Error:

Question # 2: [Marks: **10]**

You have to design a C++ **template** function rotateSquare, which takes a dynamic two dimensional square matrix, its dimensions (rows, columns) size, and a rotateValue as input. This function should rotate the outermost layer of matrix by number of times, provided in rotateValue parameter, in clockwise direction. Note the following example of 4x4 matrix with rotateValue = 3:

Matrix passed to function.

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

After call Matrix →

Outer layer rotated by 3 values

12	**8**	**4**	**0**
13	5	6	**1**
14	9	10	**2**
15	**11**	**7**	**3**

Another example of 3x3matrix with rotateValue = 1:

Matrix M.

A	B	C
H	I	D
G	F	E

After call Matrix →

Outer layer rotated by 1 value

H	**A**	**B**
G	I	**C**
F	**E**	**D**

Note: No specialization is required for this function, and rotateValue will not be greater than size.

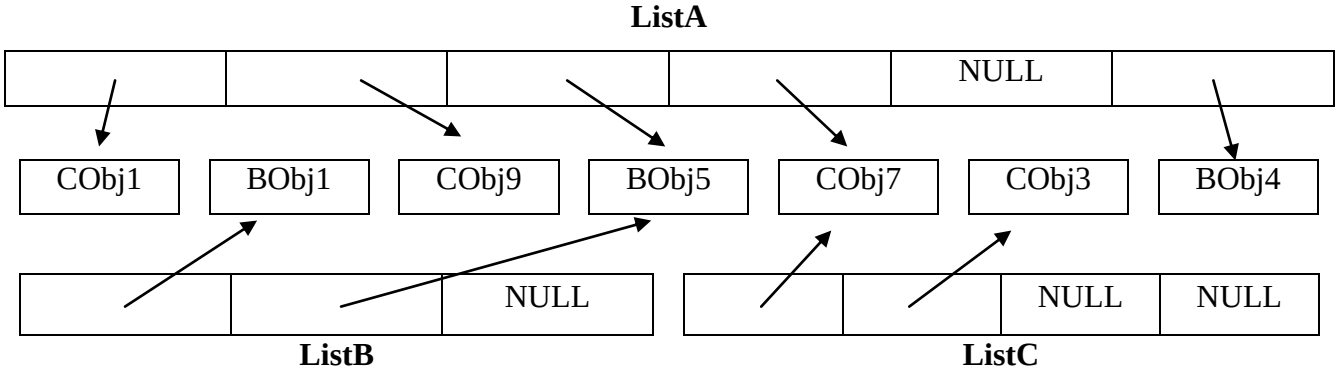
Question # 3: [Marks: **15]**

Suppose we have a class hierarchy having base class **A** and its two derived classes **B** and **C**. Definition of another class **Driver** is given:

```
class Driver
{
private:
        A** ListA;
        B** ListB;
        C** ListC;
        int countA, countB, countC;
};
```

Driver class has three arrays and their sizes. ListA, ListB and ListC are allocated on heap and their sizes are countA, countB and countC respectively. Objects of classes B and C are also on heap. ListA, ListB and ListC have pointers of these objects. Few objects of class B are shared by ListA and ListB.

Same is the case with the objects of class C. Sample Memory configuration of Heap is given below:



Write Destructor of Driver class to deallocate all this memory. (Suppose we have virtual destructors written in class A, B and C.)

Question # 4: [Marks: 20+10]

Part (A)

You have to implement an inventory application for a Medical store (Pharmacy) by using inheritance and polymorphism principles. There are two types of items, perishable, and permanent. Every item has a name (char*), and an original price (float). Every perishable item, such as medicine, injections, drips, food supplements, dry milk etc. has an expiry date and quantity (integer value e.g. 500mg). Every permanent item, such as stethoscope, BP-operates, glucometer, thermometer etc. has an age in terms of number of days passed since the time of their entry into the pharmacy to the present day. The price of every permanent item reduces by 0.02% with every passing day. The price of a perishable item remains the same before the expiry date, and becomes zero after it. Additionally, bulk items like syringes, hand gloves, are permanent items which contains two extra attributes: one is a description (char*) which is a small sentence describing the item, and the other is set (bool), which is true if the item is part of a set, and false otherwise. The price of a bulk item never changes if it is part of a set (such as a syringe could be part of a set), but it changes just like a permanent item if it is not part of a set. You may make the following assumptions:

- A fully functional class called Date is available for use, and contains a '–' (minus) operator which returns the difference between two dates, as the number of days between them.
- A global function called currentDate is also available and returns the current date.

Part (B)

Implement a class called **Pharmacy,** which will contain a list of medical items, and will enable the driver given below to compile without any errors. You cannot change the driver program at all. Your code also must not have any memory leaks.

```
int main()
{
      int itemCount = 4;
      Pharmacy ph (itemCount);    //ph has a list of 4 items here
      Item * iptr = new Permanent("glucometer",5000, 15, 11, 2016);
      // a glucometer of price 5000 and entry date 15 Nov 2016
      ph.AddItem(iptr);
      iptr = new Perishable("Panadol", 20, 16, 11, 2018,500);
      // 500mg Panadol tablet of price 20, expiry date 16 Nov 2018
      ph.AddItem(iptr);

      iptr = new Bulk_Item( "Gloves", 500, 12, 10, 2016, "Glove box for doctors" , true);
      // glovesbox of cost 500 Rs. With entry date 12 Oct 2016
      ph.AddItem(iptr);

      iptr = new Bulk_Item("5CC Syringe", 90, 15, 7, 2017, "For Injections only" , false);
      // Syring of cost 90 Rs. each with entry date 15 July 2017
      ph.AddItem(iptr);

      ph.DisplayItems();
      // this function should print complete information of items in list
      return 0;
}
```

Note: You must supply appropriate constructors, destructors, where needed, and make sure that polymorphism is used when required. You do not need to implement irrelevant/extra functions. There should not be any memory leaks or dangling pointers in your code.