National University of Computer and Emerging Sciences, Lahore Campus

Course:	**Object Oriented Programming Lab**	**Course Code:**	**CL217**
Program:	**BS (Computer Science)**	**Semester:**	**Fall 2020**
Duration:	**2 Hours 30 minutes**	**Total Marks:**	**100**
Paper Date:	**6-Feb-2021**	**Weight**	**40%**
Section:	**All**	**Page(s):**	**4**
Exam:	**Lab Final**	**Roll. No**	

Read below Instructions Carefully:

- Understanding the question statement is also part of the exam, so do not ask for any clarification. In case of any ambiguity, make suitable assumptions.
- You have to complete exam in 2.5 hrs and make submission within the same time.
- For Q1, create a folder named by your Roll number in the format 19L-9085 which should contain .h and .cpp files of First Question. Submission Path is **\\cactus\Xeon\Fall 2020\Hamna Waseem\OOP Labs\Lab Final\BCS-3X\Q1.** (X is your section name.)
- For Q2, submit .h and .cpp files on **\\cactus\Xeon\Fall 2020\Hamna Waseem\OOP Labs\Lab Final\BCS-3X\Q2.** (X is your section name.)
- All Helper Files are provided at **\\cactus\Xeon\Fall 2020\Hamna Waseem\OOP Labs\Lab Final\Helper Files**
- Submit both questions (zipped) in on Google Classroom in assignment titled as **OOP- Lab Final Submission**.
- Your code should be intended and commented properly. Use meaningful variable names. Failure to comply will result in marks deduction.
- It is your responsibility to save your code from being copied. All matching codes will be considered cheating cases. PLAGIARISM will result in forwarding of case to Disciplinary Committee and negative marking.
- In case of missing submission, Zero marks will be awarded.

Question No. 01: Marks: 60

i)

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.

ii)

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. (Main is provided on XEON Folder)

```
int main()
{
      int itemCount = 4;
      Pharmacy ph (itemCount); //ph has a list of 4 items here

      Item * iptr = new Permanent("glucometer",5000, 15, 11, 2020);
      // a glucometer of price 5000 and entry date 15 Nov 2020
      iptr->print()    //Print all information i.e. name, actual price,
      entry date, Passed days, current price.
      ph.AddItem(iptr);
      iptr = new Perishable("Panadol", 20, 16, 01, 2021,500);
      // 500mg Panadol tablet of price 20, expiry date 16 Jan 2021
      iptr->print()    //Prints name, current price(depends on expiry
                       //date), quantity and expiry date.
      ph.AddItem(iptr);

      iptr = new Bulk_Item( "Gloves", 500, 12, 10, 2020, "Glove box for doctors" ,
      true);           // glovesbox of cost 500 Rs. With entry date 12
                       //Oct 2020
      iptr->print()    //prints name, actual price, entry date, Passed
      days, current price, description and if it is part of set or not.

      ph.AddItem(iptr);

      iptr = new Bulk_Item("5CC Syringe", 90, 15, 5, 2020, "For Injections only" , false);
      // Syring of cost 90 Rs. each with entry date 15 May 2020
      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.

Question No. 02: Marks: 40

Data Structures are used in programming languages to store data. Just like C++ have arrays, Python has a data structure called **List**. Every element of list is called an **item**. Following are python lists:

```
List1 = ["apple", "banana", "cherry"]
List2 = ["abc", 34, True, 40, "male"]
```

Lists are quite similar to arrays but also have some additional features that arrays do not offer. Traditionally lists can store items of multiple data types but for now we will stick to lists with float type items only. Every list can accommodate infinite number of items i.e. there is no bound on how many items can a list hold. List's size grows dynamically as a new item is inserted in it.

Your task is to implement an amateur version of Python Lists in C++. Your version should be capable of:

1. **Insertion of new item in the list**

   To insert a new item in list, Use **+** operator.

   For Example:

   Mylist=[2.5, 0.8, 8, 5.4, 10.9]
   Mylist=Mylist+6.2
   Now Mylist=[2.5, 0.8, 8, 5.4, 10.9, 6.2]

2. **Concatenation of a list with other list**

   Insertion of multiple items should also be implemented using **+** operator

   For Example:

   Mylist1=[2.5, 0.8, 8, 5.4, 10.9]
   Mylist2=[5.0, 8.5, 6.2]
   Mylist1=Mylist1+ Mylist2
   Now Mylist=[2.5, 0.8, 8, 5.4, 10.9, 6.2, 5.0, 8.5, 6.2]

3. **Finding length of list**

   The function **length()** should return the length of list.

   For Example:

   Mylist=[2.5, 0.8, 8, 5.4, 10.9]
   cout<<Mylist.length() //output=5

4. **Pop the first element from list**

   Popping of first item should be via **–** operator.

   For Example:

   Mylist=[2.5, 0.8, 8, 5.4, 10.9]
   float item=-Mylist //Now item=2.5 and Mylist=[0.8, 8, 5.4, 10.9]

5. **Find index of an item present in the list**

   Given a float inside the **[ ]** notation, the program should be able to return its index number. If the number is not an item of list, -1 should be returned.

   For Example:

   Mylist=[2.5, 0.8, 8, 5.4, 10.9];
   cout<< Mylist[0.8]; //output will be 1

cout<< Mylist[62]; //output will be -1

6. **Reverse the list**
   Write a function **reverse()** that will reverse the element of lists without using any extra storage.
   For Example:
   if
   Mylist=[2.5, 0.8, 8, 5.4, 10.9]
   then
   Mylist.reverse(); //After reverse, MyList will be [10.9, 5.4, 8, 0.8, 2.5]

7. **Print the list**
   Overload << Operator to print the elements of list
   For Example:
   cout<<MyList;

   Output:
   [2.5, 0.8, 8, 5.4, 10.9]

Note: You must supply appropriate constructors, destructors. You do not need to implement irrelevant/extra functions. There should not be any memory leaks or dangling pointers in your code.