National University of Computer and Emerging Sciences, Lahore Campus

Course:	Computer Programming	Course Code:	CL-103
Program:	BS (Computer Science)	Semester:	Summer 2017
Duration:	150 Minutes	Total Marks:	50
Paper Date:	28-Jul-17	Weight	-
Section:	N/A	Page(s):	3
Exam:	Fianl Term	Roll No.	

Problem # 1: Please implement following classes carefully.[20 marks] Expected Time : 1 hour 10 mins

You are asked to write a class Page which has 4 types of pages listed below. There are some functions which they need to be implemented. Now there is a document which has two types in our case. These documents can have diffrent type of pages inside them. Like a Resume document can have SkillPage and ExperiencePage and some other document can have some different pages as well. But for your ease you are given with all types of pages and documents. So extract all information from these classes and their functions. As you could have to write the document [write data on respective pages], you could have to print these documents, So for them you have to write some functions which you can see below.

```
class Page{
protected:
     string title, dataOfPage;
//overload << & >> operators to take input/output in data of page.
//Every child class should have its own opertaors/functionality.
};
class SkillsPage : public Page{};
class ExperiencePage : public Page{};
class IntroductionPage : public Page{};
class TableOfContentsPage : public Page{};
// Creator
class Document{
public:
    Document(){       CreatePages();    }
    virtual void CreatePages() { };
```

```
    virtual void WriteDocument(){};//input>> in all pages from user
    virtual void PrintDocument(){ };//print<< all pages to console.
protected:
    Page* thePageList;
};
class Resume : public Document{
public:
    void CreatePages(){
        thePageList[0] =  new SkillsPage();
        thePageList[1] =  new ExperiencePage();
     }
};
class Report : public Document{
public:
    void CreatePages()    {
        thePageList[0] = new TableOfContentsPage()
        thePageList[1] = new IntroductionPage()
    }
};
void main(){
     Document* documents[2];
    documents[0] = new Resume();
    documents[1] = new Report();
//Please write Documents with good titles and data and finally
print those documents to console.       }
```

Problem # 2: [30 marks] Expected Time : 1 hour 20 minutes

We are given a stack data structure with push and pop operations [As you did in last lab], the task is to implement a queue using instances of stack data structure and operations on them. A queue can be implemented using two stacks. Let queue to be implemented be que and stacks used to implement que be stack1 and stack2.

Solution for Problem:

enQueue: 1) Push x to stack1 (assuming size of stacks is 10).

deQueue: 1) If both stacks are empty then error.

2) If stack2 is empty

While stack1 is not empty, push everything from stack1 to stack2.

3) Pop the element from stack2 and return it.

When you have implemented any one of above method you have to care new two things.

1. Any type of data can be **enqueue**d**/dequeued**. So make this que through **templates.**
2. You have initialy size 10 for both stacks, it means size of que is also 10. So whenever user tries to **enqueue** an extra element from its size, you should throw an exception with a message.
   "Exception: The Que is full right now. Do you want to flush the already existing elements....?" y/n if user answers "y", Please delete all elements form que and if users presses "n" exit the program.

   If user has **dequeued** every element and now again wants to do te same thing. You should throw an exception with following message:

   "Exception: The Que is full empty now. Do you want to exit....?" y/n

   If user enters "y", exit the program else he should be allowed to enqueue the elements again.

HINT: You can do this by taking input in some character that will make decision for **do-while loop**.

Example Code:

```
A Que class at least will look in our point of view. No compulsion for you.
template <class T>
class Que{
	Stack<T> *stack1, *stack2;
public:
	Que(){
		stack1 = new Stack<T>(), stack2 = new Stack<T>();
		//other required code if there is any...
	}
	//other functions will be here.
};
```