Assignment 3

Dated: 18-8-2022

Deadline: 23rd August, 2022

Submission instructions

- Understanding this task is entirely a part of your assessment. Please refrain from asking any question about assignment.
- Attempt the task as per your understanding
- This is group assignment.
- There can be 1-3 members in each group.
- Cheating cases will be considered strictly and might result in drop out from course.
- Submission email must have a proper subject and all your relevant submission files must be grouped or zipped to one file.
- Rename that file to roll numbers of group members. Violation in email instructions will result in marks deduction.
- Further details regarding submission will be shared by TAs soon.
- You must attempt the following task using OOP paradigm. Your solution code must have relevant classes with all necessary members and member functions.

Task

Assume you have to simulate the application of printing a list of documents using a printer. The printer maintains a queue in which it inserts the upcoming documents requests.

Design an abstract class named **Document** with document ID as a member. It should have:

- Constructor that sets the document ID
- Method named show that prints document ID and does not return anything
- Method named Dyncopy that dynamically creates and returns a copy of a document itself.

There would be two types of documents: word and PDF. Create a separate class for each type and establish appropriate relationships. **DocPDF** class should have:

- Constructor that sets the document ID.
- Method named show that prints document ID
- Method named Dyncopy that dynamically creates and returns a copy of a document itself.

DocWord should also have:

- Constructor that sets the document ID.
- Method named show that prints document ID
- Method named Dyncopy that dynamically creates and returns a copy of a document itself.

You should not allow creation of Document by itself. Create a class named **QueueP** which maintains a list of documents. You should be able to pop a document from the front and add a document to the back. Class QueueP has a linked list of nodes where each **node** consists of a document, pointer to the next node in sequence in the list. It should have:

- Constructor that sets a pdf document to a new dynamically created document and next pointer to the relevant next position.
- Constructor that sets a word document to a new dynamically created document and next pointer to the relevant next position.
- Copy Constructor that initializes the next pointer to null and document to a dynamically created word or pdf document using method Dyncopy.

Back to Class **QueueP,** it should have front and back node as data members. It should have:

- PushPDF method that takes a contant pdf document as an argument/parameter and adds it to the linked list being maintained.
- PushWord method that takes a contant pdf document as an argument/parameter and adds it to the linked list being maintained.
- Fetch method that takes a constant pointer to document as an argument. It should set it to the front node's document and return true. In case the front is NULL, it should then set the pointer to NULL and return false.
- EatFront method that deletes the current front node and made the second node as a new front and return true. In case of only one node in the list it should delete it and return true without setting the new front (as the deleted node was the only node in the list). In case the front is null i.e. the list is empty then it should then return false.
- PrintAll method that iterates the nodes in the list starting from the front and print the contents of a node.

You should also write driver main function that creates an instance of QueueP.

- Create an instance of pdf and word document.
- Add the nodes to the linked list using QueueP instance.
- Again create an instance of pdf and word document and Add the nodes to the linked list using QueueP instance
- Print the contents of QueueP's linked list
- Call EatFront and again show the contents of the changed list
- Print the contents of front node using Fetch and printAll method.