Module 11:

Pointers and Dynamic Arrays

COP2274

In-class Assignments

UF UNIVERSITY of FLORIDA

M11A Functions
(with pointers and dynamic arrays)

1. Write a function called *getNumbers()* that takes in **a pointer** to an integer variable that points to a size from the user and fills **a dynamic array** with the size from the user. The function will return **a pointer** to an integer variable that points to **a dynamic array** of that size created in the function. If the size of **a dynamic array** is equal to 0, the function will return a NULL pointer (*nullptr*).
2. Write a function called *duplicate()* that takes in an array of integers, the size of the array, and the number of copies. The function creates **a** new **dynamic array** with the capacity determined by the original array size times the number of copies. The function will return a new array with each number copied n times as shown in the test case.

M11A Functions
(with pointers and dynamic arrays)

3. Write a function called *print()* that takes in an array of integers and the size of the array and prints all the copied n times as shown in the test case.
4. Test your functions in the main() as shown in the test case.

Note:

- *Make sure to **deallocate** (delete) your **dynamic arrays** when you have finished using them!*

Test case

```
How many numbers would you like to enter? 3
Enter 3 integers
1 2 3
How many duplicated numbers do you want? 3
1 1 1 2 2 2 3 3 3
```

M11B Class called Fries
(with pointers and dynamic arrays)

1. Write a class called **Fries** with a *private* member **pointer variable** that stores **a pointer** to a variable of type double that represents the price for fries.
2. Write a *default constructor* that initializes the *private* member **pointer variable** to a NULL pointer (*nullptr*).
3. Write *a custom constructor* that takes in a double and initializes the *private* member **pointer variable** to a newly created **dynamic variable** which contains the passed double value.

M11B Class called Fries
(with pointers and dynamic arrays)

4. Write **a destructor** to *deallocate* the memory for **the dynamic variable** pointed by the *private* member **pointer variable**.
5. Write *an accessor* and *a mutator* for the value pointed to by the private member **pointer variable**. If the private member **pointer variable** is nullptr, the accessor should return 0 and the mutator should use the address (or pointer) returned from a newly created dynamic variable with the passed double value.
   *Note: The mutator should not create a new dynamic double variable unless the private member **pointer variable** is equal to nullptr.*

M11B Class called Fries
(with pointers and dynamic arrays)

6. In your main(), test your **Fries** class according to the test case.

Test case

```
The default price for fries is: $0
How much would you like to charge for fries? $4.99
Setting value...
The price of fries was set to: $4.99!
What would you like to change the price of fries to? $3.42

The price of fries has changed to $3.42!
```

M11C Classes called Client and Bank (with pointers and dynamic arrays)

1. Write **a class** called **Client** that contains two private member variables for the amount of money in checking, and in savings.
2. Write *a default constructor* for **Client** that sets the member variables to 0.
3. Write *a custom constructor* for **Client** that initializes the member variables with their corresponding parameter.
4. Write a member function of **Client** called *showData()* that prints the checking and savings balances.

M11C Classes called Client and Bank (with pointers and dynamic arrays)

5. Write **a class** called **Bank** that contains two private member variables, one for the number of clients for an array, and a pointer to a variable of type **Client** for the **dynamic array**.
6. Write *a default constructor* for **Bank** that initializes the number of clients to 0, and allocates the dynamic array with a capacity of 1.
7. Write *a destructor* for **Bank** that frees the memory for **the dynamic array** pointed by the private member pointer variable.

M11C Classes called Client and Bank (with pointers and dynamic arrays)

8. Write a public member function of **Bank** called *addClient()* which takes in a **Client** object and puts it into the **dynamic array**. Each time *addClient()* is called, reallocate the **dynamic array** with the new size. The number of clients will increase each time the function is called. *Hint: You can create another temporary **dynamic array** to assist in copying the old **dynamic array** to the new **dynamic array**.*
9. Write a public member function of **Bank** called *showData()* that prints all of the clients data by calling the *showData()* from **Client**.

M11C Classes called Client and Bank (with pointers and dynamic arrays)

10. Test your **Client** and **Bank** classes with the member functions with hardcoded values as shown in the test case.

Note:

- *You may use the **Bank** and **Client** classes from M7B to help you answer this question.*

M11C Classes called Client and Bank (with pointers and dynamic arrays)

Test case

```
After adding client 1:
Client 1:
Checking Balance: 2010.71
Savings Balance: 9876.33

After adding client 2:
Client 1:
Checking Balance: 2010.71
Savings Balance: 9876.33

Client 2:
Checking Balance: 13.71
Savings Balance: 0.00
```

```
After adding client 3:
Client 1:
Checking Balance: 2010.71
Savings Balance: 9876.33

Client 2:
Checking Balance: 13.71
Savings Balance: 0.00

Client 3:
Checking Balance: 500.00
Savings Balance: 600.00
```

```
After adding client 4:
Client 1:
Checking Balance: 2010.71
Savings Balance: 9876.33

Client 2:
Checking Balance: 13.71
Savings Balance: 0.00

Client 3:
Checking Balance: 500.00
Savings Balance: 600.00

Client 4:
Checking Balance: 9622.00
Savings Balance: 20000.00
```