CL103 COMPUTER PROGRAMMING

LAB 06

COPY CONSTRUCTOR, DESTRUCTOR AND MEMBER INITIALIZER LIST

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

COPY CONSTRUCTOR

The copy constructor initializes an object with another object of the same type. It is called automatically when a second, already existing object is used to initialize an object.
The copy constructor is used to:

- Initialize one object from another of the same type.
- Copy an object to pass it as an argument to a function.
- Copy an object to return it from a function.

SHALLOW COPYING OR MEMBER WISE COPYING

The default copy constructor and the assignment operators use the copying method known as member wise copy or shallow copy. Shallow copy means copying the data member by member using the assignment operator. The assignment operator just copies the address of the pointer but it would not allocate any memory or copy the contents being pointed to. This will work fine when the class is not dealing with the dynamically allocated memory.

PROBLEMS WITH SHALLOW COPYING

Deleting the object "student1" causes the second object "student2" to point to a deallocated memory location (i.e. dangled).
Updating the object "student1" also updates the second object "student2".

DEEP COPYING

This duplicates the member variables that are pointed to the destination. Therefore, there will be the copy of another member is created which is local to the destination object. So to implement the deep copy we need to write explicitly our copy constructor and the assignment operator functions.

DESTRUCTOR

Objects that were created by a constructor must also be cleaned up in an orderly manner. The tasks involved in cleaning up include releasing memory and closing files.
Objects are cleaned up by a special method called a *destructor,* whose name is made up of the class name preceded by ~ (tilde).

Syntax: ~class_name();

MEMBER INITIALIZER LIST

C++ provides a method for initializing class member variables (rather than assigning values to them after they are created) via a **member initializer list** (often called a "member initialization list").

The member initializer list is inserted after the constructor parameters. It begins with a colon (:), and then lists each variable to initialize along with the value for that variable separated by a comma. The need of assignments in the constructor body is replaced by member initializer list. The initializer list does not end in a semicolon. Data member are initialized in order they are declared. Order in member initializer list is not significant at all.

EXAMPLE PROGRAM (SHALLOW COPYING)

```
#include<iostream>
#include<cstring>
using namespace std;

class Student
{
	public:
		char *name;

		Student(char *nam)
		{
			int m_nLength = strlen(nam) + 1;

			// Allocate memory equal to this length
			name= new char[m_nLength];

			// Copy the parameter into name
			strcpy(name,  nam);

			// Make sure the string is terminated
			name[m_nLength-1] = '\0';
		}

		DeleteNamePtr()
		{
			delete name;
		}


		void Display()
		{
			cout<<endl<<"Name: "<<name<<endl;
		}

};

int main()
{
	Student student1("Dummy");

	Student student2 = student1;     // uses default copy constructor i.e. shallow copy

	student1.Display();
	student2.Display();

	student1.DeleteNamePtr();
	cout<<endl<<"student1 object destroyed";

	student1.Display();
	student2.Display();
	cin.get();
	return 0; }
```

```
Name: Dummy

Name: Dummy

student1 object destroyed
Name: ΞX?

Name: ΞX?


--------------------------------
Process exited after 3.602 seconds with return value 0
Press any key to continue . . .
```

EXAMPLE PROGRAM (DEEP COPYING)

```
#include<iostream>
#include<cstring>

using namespace std;

class Student
{
	public:
		char *name;

		Student(char *nam)
		{
			int m_nLength = strlen(nam) + 1;
			// Allocate memory  equal to this length
			name= new char[m_nLength];
			// Copy the parameter into name
			strcpy(name, nam);
			// Make sure the string is terminated
			name[m_nLength-1] = '\0';
		}


		Student(const Student &obj ) //COPY constructor
		{
			name = new char[strlen(obj.name)+1];
			strcpy(name,obj.name);
		}

		DeleteNamePtr ()
		{
			delete name;
		}

		void Display()
		{
			cout<<endl<<"Name: "<<name<<endl;
		}

};

int main()
{
	Student student1("Dummy");

	Student student2 = student1; // calls user defined copy constructor i.e. deep copying

	student1.Display();
	student2.Display();

	student1.DeleteNamePtr();
	cout<<endl<<"student1 object destroyed";

	student1.Display();
	student2.Display();

	cin.get();
	return 0;
}
```

```
Name: Dummy

Name: Dummy

student1 object destroyed
Name: 0§ï

Name: Dummy


--------------------------------
Process exited after 12.2 seconds with return value 0
Press any key to continue . . .
```

EXAMPLE CODE (MEMBER INITIALIZER LIST & DESTRUCTOR)

```
#include<iostream>
using namespace std;

class Account
{
private:
double balance; // Account balance

public: //Public interface:
string name; // Account holder
long accountNumber; // Account number

Account(string user, long number, double bal):name(user),accountNumber(number),balance(bal)
{

}

~ Account()
{
        cout<<endl<<this->name<<" Destroyed";
}

void displayDetails()
{
        cout<<"Account Holder: "<<name<<endl;
        cout<<"Account Number: "<<accountNumber<<endl;
        cout<<"Account Balance: "<<balance<<endl<<endl;
}

};

int main()
{
        cout<<"First statement in main"<<endl<<endl;

        Account currentAccount("Dummy1",12345,20000);
        cout<<"Details for currentAccount: "<<endl;
        currentAccount.displayDetails();

        Account savingAccount("Dummy2",678910,25000);
        cout<<"Details for savingAccount: "<<endl;
        savingAccount.displayDetails();

        Account fixedAccount("Dummy3",11121314,95000);
        cout<<"Details for fixedAccount: "<<endl;
        fixedAccount.displayDetails();

        cout<<endl<<"Last statement in main"<<endl;
        return 0;
}
```

```
First statement in main

Details for currentAccount:
Account Holder: Dummy1
Account Number: 12345
Account Balance: 20000

Details for savingAccount:
Account Holder: Dummy2
Account Number: 678910
Account Balance: 25000

Details for fixedAccount:
Account Holder: Dummy3
Account Number: 11121314
Account Balance: 95000


Last statement in main

Dummy3 Destroyed
Dummy2 Destroyed
Dummy1 Destroyed
--------------------------------
```

LAB 06 EXERCISES

INSTRUCTIONS:

NOTE: Violation of any of the following instructions may lead to the cancellation of your submission.

1) Create a folder and name it by your student id (k16-1234).
2) Paste the .cpp file for each question with the names such as Q1.cpp, Q2.cpp and so on into that folder.
3) Submit the zipped folder on slate.

QUESTION#1

Consider the code provided above for shallow and deep copy.
Create a class 'Employee' having two data members 'EmployeeName'(character pointer) and 'EmployeeId' (integer).Keep both data members private. Create an object 'Employee1' of type 'Employee'. Copy the contents of 'Employee1' to 'Employee2' in such a manner that if 'Employee1' gets deallocated or updated, 'Employee2' still holds the assigned data.
Analyze the benefits of deep copying over shallow copying.

QUESTION#2

Create a class called **String** which consists of only one data member which is a character pointer. Provide the following member functions to this class:-

1. A default constructor which initializes the pointer to null value.
2. A parameterized constructor which receives a string as its parameter. {Note:- memory allocation to be done}
3. A copy constructor which receives a string as its parameter. {Note:- memory allocation to be done}
4. A display function to display the string object.
5. A destructor to deallocate the memory which was allocated dynamically.
6. ChangeCase, which converts all lower case to upper case and vice versa.
7. Reverse, which reverses the character array.

QUESTION#3

Define a class **IntArr** which hosts an array of integers. Provide the following member functions:-
A ***default constructor.***
A ***parameterized constructor*** which initializes the array of the object.
A ***copy constructor***.
A function called ***display*** to display the array contents.
A function called ***search*** to search for an element in the array.
A function called ***compare*** which compares 2 **IntArr** objects for equality.

QUESTION#4

Define a class named **Movie**. Include private fields for the title, year, and name of the director. Include three public functions with the prototypes

```
void setTitle(char [ ]);
void setYear(int);
void setDirector(char [ ]);
```

Include another function that displays all the information about a Movie.
Include a function which accepts 2 objects of type Movie and displays whether or not they were released in the same year and also whether the Directors are same.
Provide a default constructor, a parameterized constructor and a copy constructor to this class.
Illustrate all the constructors as well as all the methods by defining objects.

QUESTION#5

Demonstrate a class that print the combination of strings. Your program should use two constructors. The first constructor should be an empty constructor which allows to declare an array of string. The second constructor initializes the length of the string, allocates the necessary space for the string to be stored and creates the string itself using strcpy. Your program should consist of one member function Join() that concatenates two strings. It should estimate the length of the string to be joined, allocates the memory for the combined string and then uses strcpy() to copy the string and strcat() to concatenate the strings. Your main() function should concatenate three strings into one string such as Mirza Arsalan Baig.

QUESTION#6

Consider a publishing company that needs a class to represent the books in stock. For this purpose, define a class called Book having the following members

Private members:

Book number: long
Book title: string
Sales price: double

Public members:

Book(long, const string&, double);
~Book();
void Display(); // Formatted output
set- and get-methods for any data member

Declare the constructor with default arguments for each parameter to ensure that a default constructor exists for the class. Negative prices must not exist. If a negative price is passed as an argument, the price must be stored as 0.0.
Implement the constructor, the destructor, and the method Display().
Define a global variable for the number of Book type objects.
The constructor must use the arguments passed to it to initialize the data
members, additionally increment the global counter, and displays the following message:

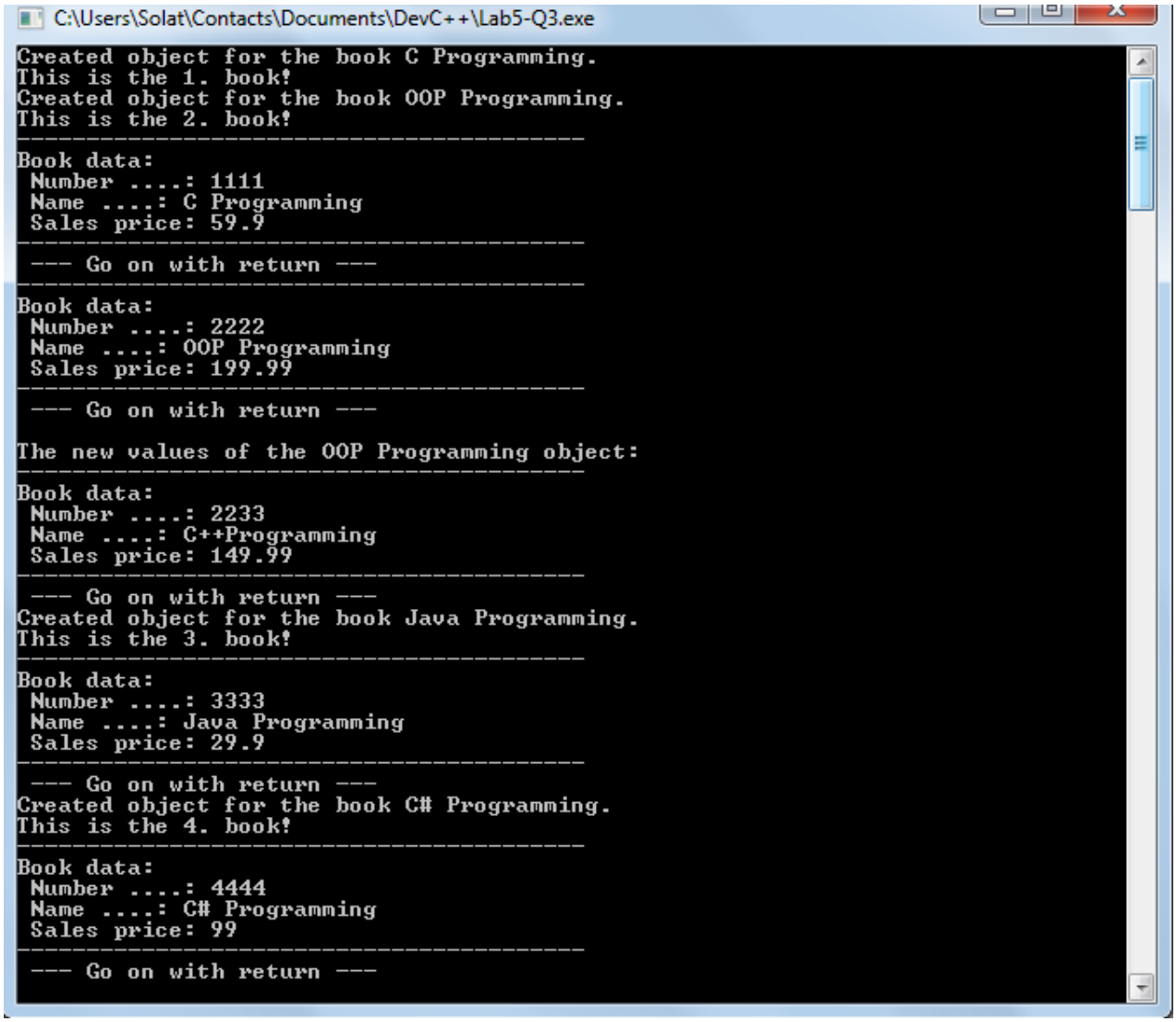
Object of class Book _____ is created.
This is the _____ Book.

The destructor should also display the following message and decrement the global counter:

Object of class Book _____ is destroyed.
There are still _____ books.

The method Display() displays a formatted object on screen. Define four objects belonging to the Book class type. Use books of your own choice to initialize the objects.
Test your program. Note the order in which constructors and destructors are called.

The following sample output should be generated for the constructor calls:


C:\Users\Solat\Contacts\Documents\DevC++\Lab5-Q3.exe
Created object for the book C Programming.
This is the 1. book!
Created object for the book OOP Programming.
This is the 2. book!
Book data:
Number ....: 1111
Name ....: C Programming
Sales price: 59.9
--- Go on with return ---
Book data:
Number ....: 2222
Name ....: OOP Programming
Sales price: 199.99
--- Go on with return ---
The new values of the OOP Programming object:
Book data:
Number ....: 2233
Name ....: C++Programming
Sales price: 149.99
--- Go on with return ---
Created object for the book Java Programming.
This is the 3. book!
Book data:
Number ....: 3333
Name ....: Java Programming
Sales price: 29.9
--- Go on with return ---
Created object for the book C# Programming.
This is the 4. book!
Book data:
Number ....: 4444
Name ....: C# Programming
Sales price: 99
--- Go on with return ---