National University of Computer & Emerging Sciences, Karachi
Spring-2022 School of Computing (BSCS, BSCY)
Midterm 2
18th April 2022, 10:00 am – 11:00 am

Course Code: CS1004	Course Name: Object Oriented Programming
Instructor Name: Dr. Farooque Hassan Kumbhar, Mr. Zain ul Hassan, Ms. Farah Sadia, Ms. Nida Munawar, Ms. Abeer Gauher, Mr. Basit Ali	
Student Roll No:	Section No:

Instructions:

- Return the question paper and make sure to keep it inside your answer sheet.
- Read questions completely before answering. There are **3 questions, 2 sides on 1 page**.
- In case of any ambiguity, you may make assumption. But your assumption should not contradict any statement in the question paper.
- You are **not allowed to write** anything on the question paper (except your ID and section).

Time: 60 minutes. **Max Marks**: 45 Marks

Q 1. Write short answers to the following questions: **[15 min, 10 Marks (2 each)]**

a. Write a code for global overloaded decrement operator **--** **(**in both prefix and postfix manner**)** to increment price of Mango class object by 100 PKR. (Assume class Mango with *price* private member).
b. Can we declare base class constructor or destructor as virtual? if yes then outline reason?
c. When and why does the diamond problem arise in inheritance?
d. What is the difference between static function, inline function, friend function, constant function?
e. What is the difference between method overloading and constructor overloading?

a) Write a code for global overloaded decrement operator -- (in both prefix and postfix manner) to increment price of Mango class object by 100 PKR. (Assume class Mango with price private member).

```
class Mango{
        float price;
        public:
                friend Mango & operator ++ (Mango & h);
                friend Mango operator ++( Mango &, int);
                void display(){
                        cout<<"Total Price are "<<price<<endl;
                }
};
Mango & operator ++ (Mango & h){ //pre increment
        h.price += 100;
        return h;
}
Mango operator ++ ( Mango & h, int){//post increment
        h.price += 100;
        return h;
}
int main (){
        Mango M1;
        M1++;
        ++M1;
        M1++;
        ++M1;
        M1.display();

}
```

b) Can we declare base class constructor or destructor as virtual if yes then why we want to do?

The constructor cannot be virtual, because when a constructor of a class is executed there is no virtual table in the memory, means no virtual pointer defined yet. So, the constructor should always be non-virtual.

Virtual destructors are useful when you might potentially delete an instance of a derived class through a pointer to base class.

c) When and Why does the diamond problem arise inheritance?

The diamond problem arises due to multiple inheritance, if hierarchical inheritance is used previously for its base classes. All the derived classes can distinguish the base class members, but if a method is being inherited to the base classes from another class which again gets inherited into the same class (diamond shape), that may create conflict in using the function from two available classes.

d) What is the difference between static function, inline function, friend function, constant function?

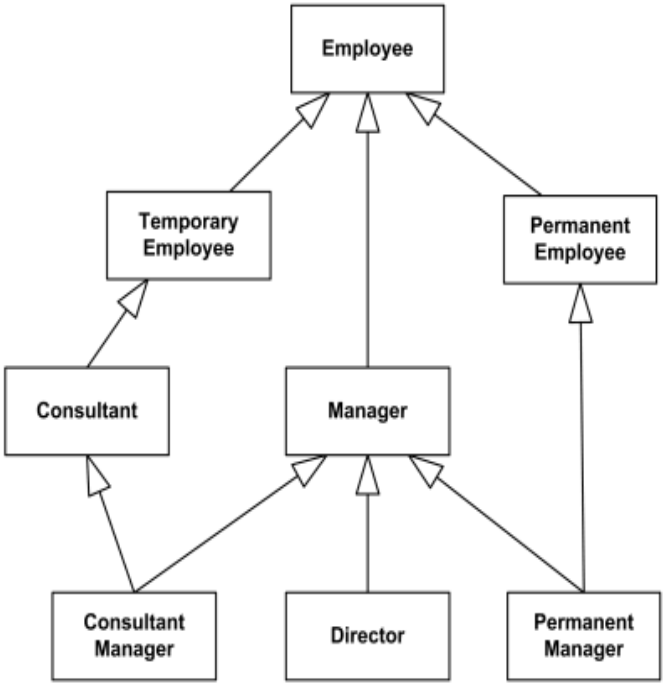
A **static function** is a member function of a class that can be called even when an object of the class is not initialized. A static function cannot access any variable of its class except for static variables.
The **inline functions** is an enhancement feature to increase the execution time of a program. Functions can be instructed to compiler to make them inline so that compiler can replace those function definition wherever those are being called. Compiler replaces the definition of inline functions at compile time instead of referring function definition at runtime.
A **friend functions** is a function that is declared outside a class but is capable of accessing the private and protected members of the class. They are defined globally outside the class scope. Friend functions are not member functions of the class.
A **const functions** is not to allow to modify the object on which they are called. It is recommended the practice to make as many functions const as possible so that accidental changes to objects are avoided.

e) What is the difference between method overloading and constructor overloading?

Constructor Overloading	Method Overloading
1. Writing more than 1 constructor in a class with a unique set of arguments is called as Constructor Overloading 2. All constructors will have the name of the class 3. Overloaded constructor will be executed at the time of instantiating an object 4. An overloaded constructor cannot be static as a constructor relates to the creation of an object 5. An overloaded constructor can be private to prevent using it for instantiating from outside of the class 6. Has no return type.	1. Writing more than one method within a class with a unique set of arguments is called as method overloading 2. All methods must share the same name 3. An overloaded method if not static can only be called after instantiating the object as per the requirement 4. Overloaded method can be static, and it can be accessed without creating an object 5. An overloaded method can be private to prevent access to call that method outside of the class 6. Has a return type.

Q 2. Consider the class diagram given below and answer the following questions: **[20 min, 15 Marks (3 each)]**

a. Provide an implementation of all the classes present in the model (declaration syntax only).
b. What will be the order of constructors when you make an object of ConsultantManager?
c. What will be the order of destructors when you destroy an object of Director?
d. Identify the type of inheritance that is involved in classes: Employee, Manager, PermanentEmployee and PermanentManager.
e. Give all possible example of multi-level inheritance from the Model.



a. Provide an implementation of all the classes present in the model (declaration syntax only).

```
class Employee { };
class TemporaryEmpolyee : public Employee { };
class PermanentEmployee : public Employee { };
class Consultant : public TemporaryEmpolyee { };
class Manager : public Employee { };
class ConsultantManager : public Consultant, public Manager { };
class Director : public Manager { };
class PermanentManager : public Manager, public PermanentEmployee { };
```

b. What will be the order of constructors when you make an object of ConsultantManager?

Employee
Temporary Employee
Consultant
Employee
Manager
Consultant Manager

c. What will be the order of destructors when you destroy an object of Director?

Director
Manager
Employee

d. Identify the type of inheritance that is involved in classes: Employee, Manager, PermanentEmployee and PermanentManager.

Hybrid Inheritance: Multiple and Multilevel

e. Give all possible example of multilevel inheritance from the Model.

1. **Empolyee -> Temporary Employee -> Consultant -> Consultant Manager**
2. **Empolyee -> Manager -> Consultant Manager**
3. **Empolyee -> Manager -> Director**
4. **Empolyee -> Manager -> Permanent Manager**
5. **Empolyee -> Permanent Employee -> Manager**

Q 3. Evento offers invitation cards printing service to its clients at a reasonable rate. They allow their clients to pick the background color, font color, texture, language to be printed on the card and also allows the client to decide if the card has a ribbon or not. In order to deliver printed cards to the client, Evento usually keeps track of client first name, contact# and address. Moreover, Evento also allows clients to view a catalog, which is only detail of ten already printed cards. You are required to perform the following tasks. **[25 min, 20 Marks (4 each)]**

a. Create classes *Evento*, *Card* and *Client* and declare all appropriate variables with correct data types in these classes.
b. Create a function to view the catalog of cards in class *Evento* such that detail of ten already printed cards are displayed. Use the prototype **void View_Catalog( );** for this function.
c. Use the following prototype to overload the function in Part (b) and out of the ten cards, display the details of only those cards with both the background and font colors that are given in parameters: **void View_Catalog (string bg_color, string fnt_color);**
d. Make a new class Digital_Evento and inherit it from Evento. Override both functions, from Part (b) and (c), in this child class such that the overriding functions asks the user the total number of cards to show from the catalog of ten cards before displaying.
e. Overload the comparison operator **= =** in *Evento* class such that it returns a boolean value indicating if the language of given card is the same as that of any given language.

Example expression: *bool result = (obj_card = = "english");*

Part a:

```
class Evento
{
protected:
	Card* cards;
};

class Card
{
	string bg_color, font_color, texture, language;
	bool contains_ribbon;
};

class Client
{
	string first_name, contact, address;
};
```

Part b:

```
class Evento
{
protected:
	Card* cards;
public:
	Evento()
	{
		cards = new Card[10];
		for (int i = 0; i < 10; i++)
		{
			// create card objects and save in array
		}
	}

	void view_catalog()
```

```
	{
		for (int i = 0; i < 10; i++)
		{
			cards[i].print_details();
			// print_details displays all attributes
		}
	}
};
```

Part c:

```
void view_catalog(string bg_color, string fnt_color)
{
	for (int i = 0; i < 10; i++)
	{
	if(cards[i].get_bgcolor() == bg_color && cards[i].get_fontcolor() == fnt_color)
		cards[i].print_details();
		// print_details displays all attributes of selected cards
	}
}
```

Part d:

```
class Digital_Evento: public Evento
{
void view_catalog()
{
	int n = 0;
	cout << "Enter how many cards to display" << endl;
	cin >> n;
	for (int i = 0; i < n; i++)
	{
		cards[i].print_details();
		// print_details displays all attributes
	}
}

void view_catalog(string bg_color, string fnt_color)
{
	int n = 0;
	cout << "Enter how many cards to display" << endl;
	cin >> n;
	for (int i = 0; i < n; i++)
	{
	if (cards[i].get_bgcolor() == bg_color && cards[i].get_fontcolor() == fnt_color)
		cards[i].print_details();
		// print_details displays all attributes of selected cards
	}
}

};
```

Part e:

```
bool operator==(string language)
{
	int i;
	cout << "Enter card to compare language" << endl;
	cin >> i;
	if (this->cards[i].get_language() == language)
		return true;
	return false; // else: return false
}
```

BEST OF LUCK!