National University of Computer & Emerging Sciences, Karachi
Spring-2023 School of Computing (BSCS, BSCS-R, BSSE, BSCY, BSAI)
Midterm 1 SOLUTION
27th Feb 2023, 10:00 am – 11:00 am

Course Code: CS1004	Course Name: Object Oriented Programming

Q 1. Write two-line short answers to the following questions: **[10 min, 6 Marks]**

a. If we add a parameterized constructor, then there is no need for a setter function. Do you agree with this? explain.

No, we can initialize the data inside the constructor but what if we need to update the value afterwards so we can change it. That's why we need a setter function.

b. How can you prevent copying of objects?

Creating private copy constructor

c. Why is "this" operator used? Is it possible to use "this" operator outside of a class? Defend.

This operator is used for:

1. **When local variable's name is same as member's name**
2. **To return reference to the calling object (chaining)**
3. **No**

d. Can static functions access non-static data? Defend your answer.

No, static function cannot access the non-static data

e. What are the cases in which the copy constructor of a class gets called?

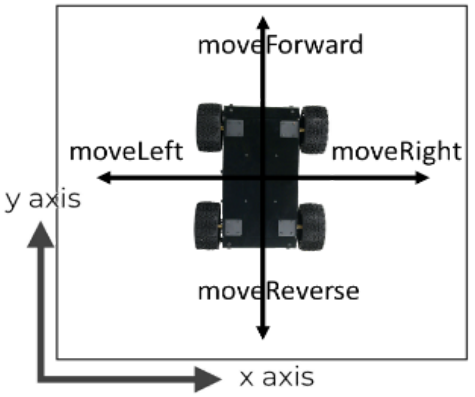
1. **When you pass an object explicitly as an argument.**
2. **When copying object using assignment operator**

f. Why is a constructor automatically called when an object is created in OOP?

When you want to initialize the value at the time of object creation.

Q 2.

In the world of AI, let's make a robot that cleans the house by taking x, y position. A robot has information of its current position as an x-axis point (left and right) and y-axis point (forward and reverse). The Robot can move Left, Right, Forward and Backwards. For this purpose, the robot has the following methods: **[25 min, 6+6 Marks]**



moveLeft that moves the robot from the current position to the left for '**d**' distance.
moveRight that moves the robot from the current position to the right for '**d**' distance.
moveForward that moves the robot forward for '**d**' distance.
moveReverse that moves the robot backwards for '**d**' distance.

A. Write a program that includes Robot class with appropriate variables and functions. Do include default and parameterized constructors with default values. Must include a copy constructor that can be used to copy values of an object and assigned to another. Just for the fun of it, copy constructor creates an opposite image of existing object, i.e.

assignment of x to y and y to x. Depending on the variable type and code requirements, define accessor/ mutator functions.

B. Write a member function void commands (string act, int d) that moves the robot by reading each character from the string. For example, act = "RRLF" then there are four actions, and the actions will move robot in right direction with d points, then right direction with d points, then left direction with d points, then forward direction with d points. Following is the character-wise action: [L for moveLeft, R for moveRight, F for moveForward, B for moveReverse].

Q2 Solution:

```
#include <iostream>

using namespace std;

class Robot {

private:

   int x, y; // current position of the robot

public:

   // default constructor

   Robot() {

      x = y = 0;

   }

   // parameterized constructor

   Robot(int x_pos, int y_pos) {

      x = x_pos;

      y = y_pos;

   }

   // copy constructor

   Robot(const Robot& other) {

      x = other.y;

      y = other.x;

   }

   // accessor functions
```

```
int getX() const {
    return x;
}
int getY() const {
    return y;
}
// mutator functions
void setX(int x_pos) {
    x = x_pos;
}
void setY(int y_pos) {
    y = y_pos;
}
// move robot left
void moveLeft(int d) {
    x -= d;
}
// move robot right
void moveRight(int d) {
    x += d;
}
// move robot forward
void moveForward(int d) {
    y += d;
}
// move robot backwards
void moveReverse(int d) {
    y -= d;
```

```
}
// execute commands from a string
void commands(string act, int d)
      {
            int i;
            char c;
            for(i=0;i<act.length();i++)
            {
                  c=act.at(i);
      switch(c) {
         case 'L':
            moveLeft(d);
            break;
         case 'R':
            moveRight(d);
            break;
         case 'F':
            moveForward(d);
            break;
         case 'B':
            moveReverse(d);
            break;
         default:
            cout << "Invalid command: " << c << endl;
      }
   }
   }
```

```
    void display()
    {
        cout<<"The postions of the robot are: "<<x<<" "<<y;
        }
};
int main() {
    // create a robot object
    int x,y,d;
        cout<<"Input the initial x,y position of the robot ";
    cin>>x>>y;
    Robot r1(x, y);//2,3
        int a=r1.getX();
        int b=r1.getY();
        cout<<"Input the d position of the robot ";
    cin>>d;
    r1.commands("RRLF", d);//2
        cout << "Initial position: (" << a << ", " << b << ")" << endl;
    cout << "Final position: (" << r1.getX() << ", " << r1.getY() << ")" << endl;
    // create another robot object using copy constructor
    Robot r2 = r1;
    // print position of the new robot object
     cout << "Position of r2: (" << r2.getX() << ", " << r2.getY() << ")" << endl;
    return 0;
}
```

Q3: "Netflix2" animation studio produces animated movies and aims to properly manage their "movie projects" and "staff". The studio works on many different movies projects and staff assigned to them. Write a program to perform the given tasks: **[25 min, 5+5+2 Marks]**

A. To maintain information of a movie project. The movie information includes movie id, title, the total budget, and current cost of the movie. Moreover, each movie has two staff members, a project lead and chief animator. Apart from this, the movie project also has the following behavior: **void Production (),** This behavior allows project lead and chief animator to perform their assigned job and print that the task is in production. The class should have a global member that keeps track of total budget of all the movies created.

B. Each staff information includes staffID, salary, employee type. Employee type can have only two possible values of project lead or chief animator. The class should have a global member that keeps track of total salaries of all the staff employed. Apart from this, an employee demonstrates the behavior **double TrackProject (double currentCost)** if the type of that employee is project lead. And **double Animate (double currentCost)** if the type of that employee is chief animator.
   i. Each time the chief animator animates, it adds PKR 10,000 to the current cost of the movie.
   ii. When a project lead tracks a project, he checks if the current cost of the movie is exceeding PKR 200,000. If it is exceeding budget, then he immediately shows a "warning: that cost is exceeding budget".

C. Draw a UML / Class diagram that represents the above given system to highlight OOP features.

Q3 Solution:

// Part A and B

```
class Movie
{
        Staff ProjectLead;
                Staff ChiefAnimator;
                const double budget;
                double CurrentCost;
                static double totalBudget;
        string MovieID;
                string MovieTitle;
        public:
                Movie (Staff pl, Staff ca, double ct, double budget, string id, string title) : budget(budget)
                {
                CurrentCost = ct;
                ProjectLead = pl;
                ChiefAnimator = ca;
                MovieID = id;
                MovieTitle = title;
                totalBudget += budget;
        }
        void Production ()
        {
                cout << "Task is in production";
                ProjectLead.TrackProject (CurrentCost);
                ChiefAnimator.Animate (CurrentCost);
```

```
        } };
        double Movie::totalBudget = 0;
class Staff
{
        double salary;
        string type;
        static double totalSalary;

        public:
                Staff (double salary, string type)
                {
                        this->salary = salary;
                        this->type = type;
                        totalSalary += salary;
        }
        double TrackProject (double currCost)
                {
                        if (currCost > 200000)
                                cout << "Warning: The cost is exceeding budget."; return currCost;
        }
        double Animate (double currCost)
        {
                // animates movie
                currCost +=  10000; return currCost;
        } };
double Staff::totalSalary = 0;
```

// Part C