# LAB TASK WEEK-6

ALI ZIA KHAN

FESE-19052

**Q #1:** You are already familiar with the `Rectangle` and `DormRoom` object types and have also written class descriptions for the two classes. Write down C++ class declarations for both the class.

```
#include<iostream>
#include<conio.h>
using namespace std;
class rectangle{
    protected:
        int length;
        int breadth;
    public:
        rectangle()
        {}
        rectangle(int l, int b)
        {
            length = l;
            breadth = b;
        }

        int getlength()
        {
        return length;
        }
        int getbreadth()
        {
        return breadth;
        }

        void setlength(int l)
        {
        length = l;
        }
        void setbreadth(int b)
        {
        breadth = b;
        }
```

```
        double area(){
            return length*breadth;
        }

        virtual void showData(){
            cout << "Length = " << getlength() << endl;
            cout << "Breadth = " << getbreadth() << endl;
        }
};

class dormroom:public rectangle{
    private:
        int height;
    public:
        dormroom()
        {}
        dormroom(int l, int b, int h):rectangle(l,b), height(h)
        {}

        int getheight(){
        return height;
        }
        void setheight(int h){
        height = h;}

        double volume(){
            return length*breadth*height;
        }

        virtual void showData(){
            rectangle::showData();
            cout << "Height = " << getheight << endl;
        }
};
```

**Q #2:** A company called Restaurant Products, Inc. wants to produce small machines for use by waiters in a restaurant. Each machine is initialized with the name of the restaurant and the local sales tax rate. When a customer has finished eating, the waiter enters the cost of the meal and has the machine output a bill in the following form:

The bill automatically adds a 15% tip to the cost. After this the waiter enters the amount paid by the customer and outputs a receipt that includes the following information in the following form:

To produce the software for this machine write down the class description and declaration for the `MealBill` class.

```cpp
#include<iostream>
#include<conio.h>
#include<string.h>
using namespace std;
class Mealbills {
public:
    string resname;
    double localtax;
    double mealcost;
    double tip;
    double totalcost;
    double payment;
    double change;
public:
    Mealbills() {

    };
    Mealbills(string r, double t) {
        resname = r;
        localtax = t;

    };

    string getresname() {
        return resname;
    }


    double gettotalcost() {
        return totalcost;
    }


    void valterenterinfo() {
        cout << "ENTER COST OF MEAL ";
```

```
    cin >> mealcost;



}
void generatebill() {

    tip = mealcost * 0.15;
    cout << "mealcost\t" << mealcost << endl;
    cout << "tax:\t" << localtax << endl;
    cout << "Tip:\t" << tip << endl;
    totalcost = mealcost + (tip * 0.15) + localtax;
    cout << "Total Cost" << totalcost << endl;
}
void setpayment() {
    cout << "Enter Payment:" << endl;
    cin >> payment;

}


void finalreceipt() {
    cout << "Toatal Cost\t" << totalcost << endl;
    cout << "Payment\t" << payment << endl;
    if (payment > totalcost) {
        change = payment - totalcost;
    }
    cout << "Change\t" << change << endl;
//  cout << "Thanks for dining at\t" << resname << endl;
}


};
```

**Q #3:** Write a program for The University Summit Restaurant, which allows a student Ahmed Ali to charge the cost of a meal to his student account. Records are stored in the student account object ahmedAli that has a current balance of Rs.2000.00. The Summit Restaurant implements billing by creating a `MealBill` object, which is set to charge 6% sales tax.

The program first prompts for the cost of the meal and then computes and prints the bill. It then makes a charge to the student's account and prints the receipt. In the end the program prints the current status of Ahmed's account.

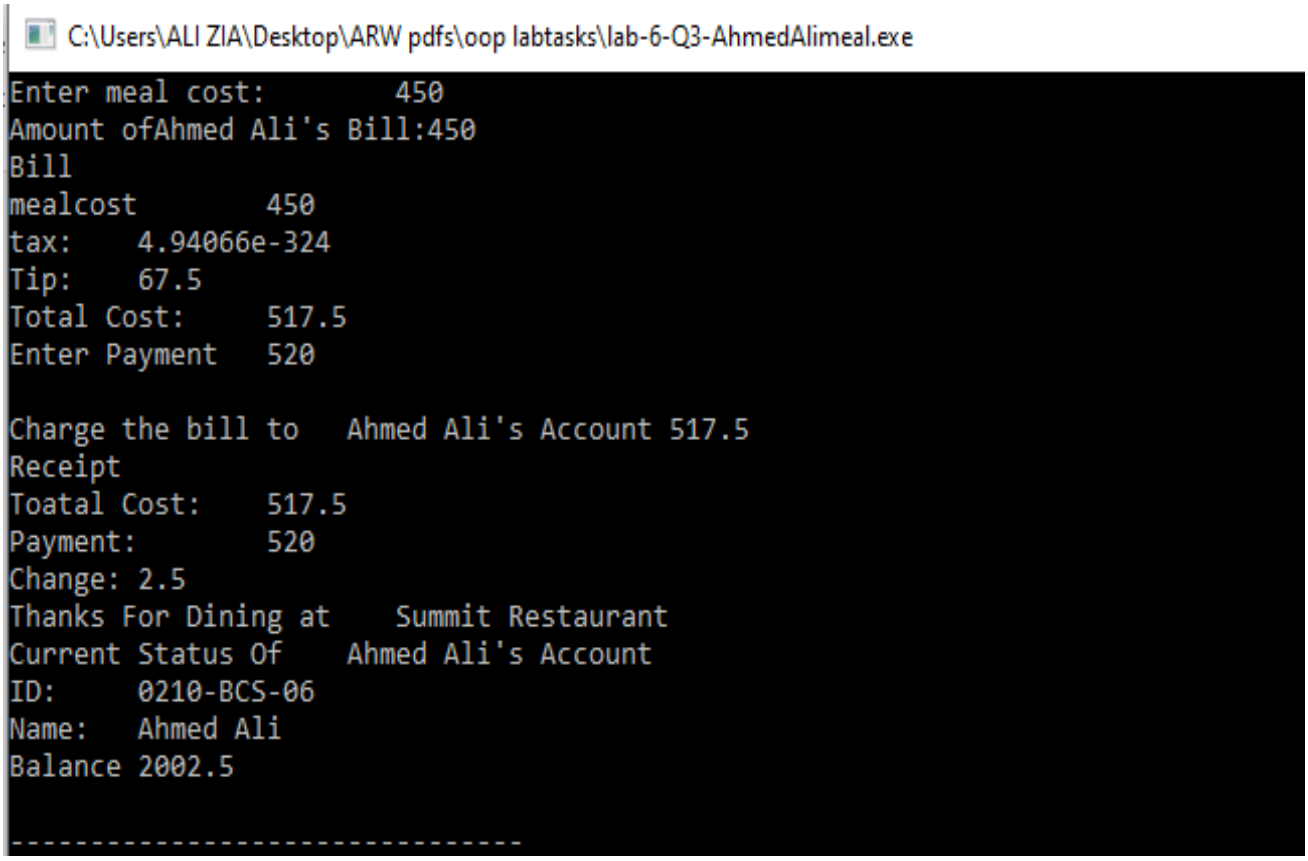
```cpp
#include<iostream>
#include<conio.h>
#include<string.h>
using namespace std;
class Mealbills {
public:
    string resname;
    double localtax;
    double mealcost;
    double tip;
    double totalcost;
    double payment;
    double change;
public:
    Mealbills() {

    };
    Mealbills(string r, double t) {
        resname = r;
        localtax = t;

    };

    string getresname() {
        return resname;
    }


    double gettotalcost() {
        return totalcost;
    }


    void valterenterinfo() {
        cout << "ENTER COST OF MEAL ";
```

```
    cin >> mealcost;



}
void generatebill() {

    tip = mealcost * 0.15;
    cout << "mealcost\t" << mealcost << endl;
    cout << "tax:\t" << localtax << endl;
    cout << "Tip:\t" << tip << endl;
    totalcost = mealcost + (tip * 0.15) + localtax;
    cout << "Total Cost" << totalcost << endl;
}
void setpayment() {
    cout << "Enter Payment:" << endl;
    cin >> payment;

}


void finalreceipt() {
    cout << "Toatal Cost\t" << totalcost << endl;
    cout << "Payment\t" << payment << endl;
    if (payment > totalcost) {
        change = payment - totalcost;
    }
    cout << "Change\t" << change << endl;
//  cout << "Thanks for dining at\t" << resname << endl;
}


};
```

```
class Account :public Mealbills {
public:
    string StudentID;
    string name;
    int balance;
public:
    Account() {

    };
    Account(string i, string n, long b)  {
        StudentID = i;
        name = n;
        balance = b;
    };
    string getname() {
        return name;
    }
    virtual void valterenterinfo() {
        cout << "Enter meal cost:\t" ;
        cin >> mealcost;


    }
    virtual void generatebill() {


        tip = mealcost * 0.15;
        localtax=localtax*0.6;
        cout<<"Bill"<<endl;
        cout << "mealcost\t" << mealcost << endl;
        cout << "tax:\t" << localtax << endl;
        cout << "Tip:\t" << tip << endl;
        totalcost = mealcost + (tip) + localtax;
        cout << "Total Cost:\t" << totalcost << endl;
    }
```

```
    virtual void setpayment() {
        cout << "Enter Payment\t";
        cin >> payment;
    }
   virtual string getresname() 
        return resname;
    }
    virtual void finalreceipt() {
        cout<<"Receipt"<<endl;
        cout << "Toatal Cost:\t" << totalcost << endl;
        cout << "Payment:\t" << payment << endl;
        if (payment > totalcost) {
            change = payment - totalcost;
        }
        cout << "Change:\t" << change << endl;      }
    void showinfo() {
        cout << "ID:\t" << StudentID << endl;
        cout << "Name:\t" << name << endl;
        cout << "Balance\t" << balance+payment- totalcost << endl;
    }
};
int main() {
    Mealbills m1("Summit Restaurant",6);
    Account ahmedAli("0210-BCS-06", "Ahmed Ali", 2000);
    ahmedAli.valterenterinfo();
    cout << "Amount of" << ahmedAli.getname() << "\'s Bill:" << ahmedAli.mealcost<<endl;
    ahmedAli.generatebill();
    ahmedAli.setpayment();
    cout << "\nCharge the bill to   " << ahmedAli.getname() << "\'s Account " <<ahmedAli.totalcost<<endl;
    ahmedAli.finalreceipt();
    cout<<"Thanks For Dining at\t"<<m1.getresname()<<endl;
    cout << "Current Status Of    " << ahmedAli.getname() << "\'s Account" << endl;
    ahmedAli.showinfo();
    return 0;
}
```

**OUTPUT:**




**Q #4:** A student's grade record is maintained by the registrar in the Academic Dept. The record includes the studentID along with the total number of credits attempted and the total grade points earned by the student. The grades points are determined by the scale A(4), B(3), C(2), D(1), and F(0).
E.g. `4-credit course with grade B: grade points = 4 * 3 = 12`

A student's grade record is initialized with his/her ID, number of credits and the grade points earned. The number of credits and grade points default to 0 (GPA 0.0), for anew student. The student record is used for the purpose of

**(i)** determining the GPA – the total credits and the grade points are used to compute the GPA.
**(ii)** to write the grade point information in the following format:
**(iii)** Each semester the registrar updates the student records with grades from recently completed courses. For instance, assume student "0210-BCS-01" completed a semester in which he/she earned 16 credits and 58 grade points. So, credits and gradePts are now increased to 116 and 403, respectively.
**(iv)**
**(v)** Write down the description and declaration for the GradeRecord class

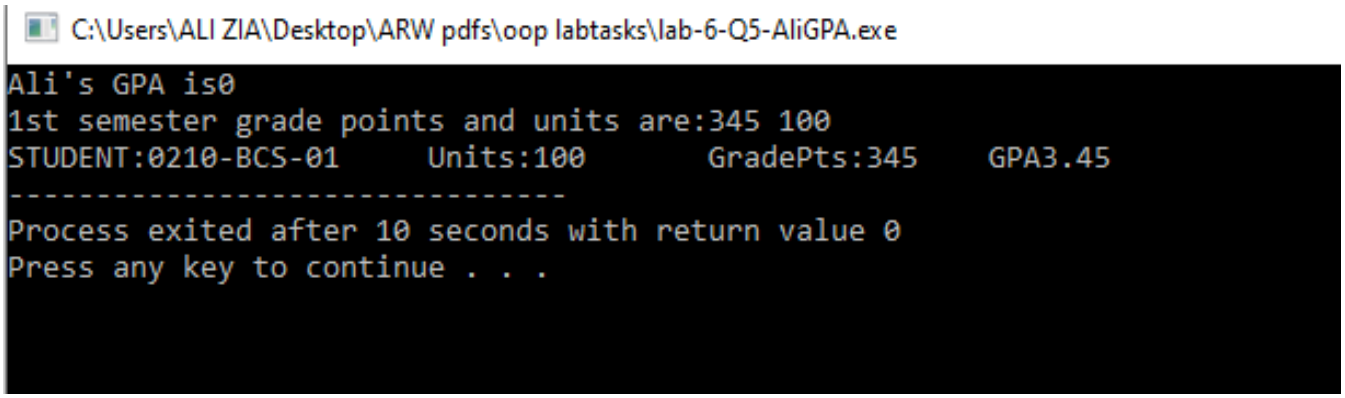
```
#include<conio.h>
#include<string.h>
#include<iostream>
using namespace std;
class GradeRecord {
    public:
        string studentID;
        int credit;
        int gradepoint;
    public:
    GradeRecord(string i){
        studentID=i;
        credit=0;
        gradepoint=0;
    };
    GradeRecord(string i, int c,int gr_p){
        studentID=i;
        credit=c;
        gradepoint=gr_p;
    };
    double calculate_GPA(){
        if(credit){

        return (double)gradepoint/(double)credit;
        }
        else
        return 0;
    }
     void gp_info(){
         cout<<"STUDENT:"<<studentID;
         cout<<"\tUnits:"<<credit;
         cout<<"\tGradePts:"<<gradepoint;
         cout<<"\tGPA";
         printf("%.2f",calculate_GPA());
     }
     void updaterec(int c,int gr_p){
         credit+=c;
         gradepoint+=gr_p;
     }
};
```

**Q #5:** Write a program that creates a student grade record for a new student. To indicate that student is new print the students initial grade point average. The program then prompts for the grade points and credits that the student has earned in the first semester and updates his/her grade record. The student's record at the end of the semester is then output to the screen.

```
#include<conio.h>
#include<string.h>
#include<iostream>
using namespace std;
class GradeRecord {
    public:
        string studentID;
        int credit;
        int gradepoint;
    public:
    GradeRecord(string i){
        studentID=i;
        credit=0;
        gradepoint=0;
    };
    GradeRecord(string i, int c,int gr_p){
        studentID=i;
        credit=c;
        gradepoint=gr_p;
    };
    double calculate_GPA(){
        if(credit){

        return (double)gradepoint/(double)credit;
        }
        else
        return 0;
    }
```

```
    void gp_info(){
        cout<<"STUDENT:"<<studentID;
        cout<<"\tUnits:"<<credit;
        cout<<"\tGradePts:"<<gradepoint;
        cout<<"\tGPA";
        printf("%.2f",calculate_GPA());
    }
    void updaterec(int c,int gr_p){
        credit+=c;
        gradepoint+=gr_p;
    }
};


int main(){

    GradeRecord Ali("0210-BCS-01");
    cout<<"Ali's GPA is"<<Ali.calculate_GPA()<<endl;
    int c,g;
    cout<<"1st semester grade points and units are:";
    cin>>g>>c;
    Ali.updaterec(c,g);
    Ali.gp_info();
    return 0;



}
```

OUTPUT:



Q6:Create a Base Class Person having attributes( name,age,gender) with behavior of showdata(displaying all the members) and override same method in derived classes.

Derive student Class and from student class derive graduate student with appropriate data members and behavior.

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

class Person {
protected:
    string name;
    string gender;
    int age;

public:
    Person() {
        name = 'NULL';
        gender = 'NULL';
        age = 0;
    }
    Person(string n, string g, int a) :name(n), gender(g), age(a)
    {
    }

    string getname()
    { return name;
     }
    string getgender() {
    return gender;
    }
    int getage() {
    return age;
    }

    void setname(string n) {
    name = n;
    }
```

```
    string getgender() {
    return gender;
    }
    int getage() {
    return age;
    }

    void setname(string n) {
    name = n;
    }
    void setgender(string g) {
    gender = g;
    }
    void setage(int a) {
     age = a;
     }

    virtual void showinfo() {
        cout << "Name : " << name << endl;
        cout << "Gender : " << gender << endl;
        cout << "Age : " << age << endl;
    }
};

class Student :public Person {
protected:
    int rollNo;

public:
    Student() {
    }
    Student(string n, string g, int a, int r) :Person(n, g, a), rollNo(r)
    {
    }
```

```
    return rollNo;
    }

    void setrollNo(int r) {
    rollNo = r;
     }

    virtual void showinfo() {
        Person::showinfo();
        cout << "Roll No : " << rollNo << endl;
    }
};

class Graduate :public Student {
private:
    double gradePoints;
    int creditHours;

public:
    Graduate() {}
    Graduate(string n, string g, int a, int r, double gp, int ch) :Student(n, g, a, r), gradePoints(gp), creditHours(ch)
    {}

    int getcredit() {
    return creditHours;

    }
    double getgrade() {
    return gradePoints;
    }

    void setcredit(int ch) {
    creditHours = ch;
    }
```

```
    void setgrade(double gp) {
    gradePoints = gp;
    }

    double cgpa() {
        return gradePoints / (double)creditHours;
    }

    virtual void showinfo() {
        Student::showinfo();
        cout << "Credit Hours : " << creditHours << endl;
        cout << "Grade Points : " << gradePoints << endl;
        cout << "GPA : " << setprecision(3) << cgpa() << endl;
    }
};

int main() {
    Graduate g1("Ali", "Male", 50, 20, 55, 19);
    g1.showinfo();

    return 0;
}
```

**OUTPUT:**

```
Name : Ali
Gender : Male
Age : 50
Roll No : 20
Credit Hours : 19
Grade Points : 55
GPA : 2.89


--------------------------------
```