OBJECT ORIENTED PROGRAMMING

LAB # 3

ASSIGNMENT

NAME: S.M.IRTIZA
ROLL NO. : 22K-4638
CLASS: 2-F

QUESTION # 01:

Define a class to represent a Employee. Include the following members

Data Members

Name of the employee
ID number
Department name
Tasks assigned
Time allocated for each task
Tasks completed in how many minutes

Member Function

Write setter and getter property to set all attributes To display name and ID number, tasks assigned and tasks completed (in minutes) by each employee .(MAX enter 5 employees)

CODE:

```
//ROLL NO.: 22K-4638 class: 2-f
#include<iostream>
#include<string>
using namespace std;
class student {
string name;
string ID;
string dept_name;
int task_assign;
int time_for_each_task;
float completion_time;
public:
void SetName(string a) {
name = a;
}
string getName() {
return name;
}
void SetID(string b) {
ID = b;
}
string getID() {
return ID;
}
void SetDeptName(string c) {
dept_name = c;
}
string getDeptName() {
return dept_name;
}
void SetTaskAssign(int d) {
task_assign = d;
}
int getTaskAssign() {
return task_assign;
}
```

```
void SetTime_For_Each_Task(int e) {
time_for_each_task = e;
}
int getTime_For_Each_Task() {
return time_for_each_task;
}
void SetCompletionTime(float f) {
completion_time = f;
}
float getCompletionTime() {
return completion_time;
}
void display() {
cout <<"name: "<< getName() << endl;
cout <<"ID: "<< getID() << endl;
cout <<"DEPT name: "<< getDeptName() << endl;
cout <<"task assign:"<< getTaskAssign() << endl;
cout <<"time for each task: "<< getTime_For_Each_Task() << endl;
cout <<"completion time: "<< getCompletionTime() << endl;
}
};
int main() {
cout<<"ROLL NO.: 22K-4638 class: 2-f"<<endl;
student s[3];
int i, n, t;
float C;
string NAME, id, DEPT_NAME;
for (i = 0; i < 3; i++) {
cout << "enter name: " << endl;
cin >> NAME;
s[i].SetName(NAME);
cout << "enter id: " << endl;
cin >> id;
s[i].SetID(id);
cout << "enter dept name: " << endl;
cin >> DEPT_NAME;
s[i].SetDeptName(DEPT_NAME);

cout << "enter number of task: " << endl;
cin >> n;
s[i].SetTaskAssign(n);
cout << "enter time for each task: " << endl;
cin >> t;
s[i].SetTime_For_Each_Task(t);
cout << "enter completion time: " << endl;
cin >> C;
s[i].SetCompletionTime(C);
}
for (i = 0; i < 3; i++) {
```

```
s[i].display();
}
return 0;
}
```

OUTPUT:

```
ROLL NO.: 22K-4638  class: 2-f
enter name:
irtiza
enter id:
22k-4638
enter dept name:
cs
enter number of task:
2
enter time for each task:
20
enter completion time:
18
enter name:
sadiq
enter id:
22k-1234
enter dept name:
cs
enter number of task:
3
enter time for each task:
30
enter completion time:
25
enter name:
subhan
enter id:
22k-3434
enter dept name:
EE
enter number of task:
4
enter time for each task:
30
enter completion time:
22
name: irtiza
ID: 22k-4638
DEPT name: cs
task assign:2
time for each task: 20
completion time: 18
name: sadiq
ID: 22k-1234
DEPT name: cs
task assign:3
time for each task: 30
completion time: 25
name: subhan
ID: 22k-3434
DEPT name: EE
task assign:4
time for each task: 30
completion time: 22
```

QUESTION # 02:

Let us design a class bankAccount. A bank account has an account number. The bank gives each account a different, unique number. Each instance of this class maintains one account with an wner, an account number and current balance. Normally, the account numbers start with some +ve integer and keep on increasing as the new accounts are created. We need a way to assign a new account number to each instance as it is created. A new account can be created by giving the owner's name and an initial amount. Nobody should be able to manipulate instance variables directly. Methods must be provided to access (i) name of the owner (ii) account number (iii) current balance, and (iv) deposit money in the account.

CODE:

```
//ROLL NO.: 22K-4638 class: 2-f
#include<iostream>
#include<string>
using namespace std;
class BankAccount {
private:
string name;
static int account_no;
int curr_bal=0;
int dep_money=0;
int num;
public:
BankAccount(){}
BankAccount(string n, int d=0) {
setname(n);
curr_bal=d;
//setdep_money(d);
account_no++;
num = account_no;
}

void setname(string a) {
name = a;
}

string getname() {
return name;
}

void setdep_money(int x=0) {
dep_money = x;
}

int getdep_money() {
return dep_money;
}

void setcurr_bal() {
```

```
curr_bal += getdep_money();
}

int getcurr_bal() {
return (curr_bal + dep_money);
}
void display() {
cout << "name: " << getname() << endl;
cout << "account number: " << num << endl;
cout << "current balance: " << getcurr_bal() << endl;
cout << "depoite money: " << getdep_money() << endl;
}
};
int BankAccount::account_no(0);

int main() {
cout<<"ROLL NO.: 22K-4638 class: 2-f"<<endl;
BankAccount b1("irtiza", 1200), b2("sadiq", 1100);
b1.display();
b2.display();
cout<<endl;
//depositing amount
cout<<"after depositing money : "<<endl;
b1.setdep_money(5000);
b1.display();
b2.setdep_money(10000);
b2.display();
return 0;
}
```

OUTPUT:

```
ROLL NO.: 22K-4638  class: 2-f
name: irtiza
account number: 1
current balance: 1200
depoite money: 0
name: sadiq
account number: 2
current balance: 1100
depoite money: 0

after depositing money :
name: irtiza
account number: 1
current balance: 6200
depoite money: 5000
name: sadiq
account number: 2
current balance: 11100
depoite money: 10000
```

QUESTION # 03:

Write Program to find the Factorial of a Number Using Recursion.Create a class called Bank Employee that includes three pieces of information as data members—a first name (type char* (DMA)), a last name (type string) and a monthly salary (type int). Your class should have a setter function that initializes the three data members. Provide a getter function for each data member. If the monthly salary is not positive, set it to 0. Write a test program that demonstrates class Employee's capabilities. Create five Employee objects and display each object's yearly salary. Then give each Employee a 20 percent raise and display each Employee's yearly salary again. Identify and add any other related functions to achieve the said goal

CODE:

```
//ROLL NO.: 22K-4638  class: 2-f
#include<iostream>
#include<string>
using namespace std;

class Bank_Employee {
private:
      char *f_name = new char[50];
      string l_name;
      int monthly_salary;
public:
    Bank_Employee(){

    }
      Bank_Employee(char *first, string y, int z) {
            setf_name(first);
            setl_name(y);
            setsalary(z);
      }

      void setf_name(char *first) {
            f_name = first;
      }
      char* getf_name() {
            return f_name;
      }

      void setl_name(string l) {
            l_name = l;
      }
      string getl_name() {
            return l_name;
      }

      void setsalary(int s) {
            if (s > 0) {
                  monthly_salary = s;
            }
            else {
                  monthly_salary = 0;
            }

      }
      int getsalary() {
            return monthly_salary;
      }
```

```
	void display() {
		cout << "f_name: " << getf_name() << endl;
		cout << "l_name: " << getl_name() << endl;
		cout << "annual salary: " << getsalary() * 12 << endl;

	}

	int increament() {
		return (getsalary() * 0.2 * 12 + getsalary() * 12);
	}

	void DISPLAY() {
		cout << "f_name: " << getf_name() << endl;
		cout << "l_name: " << getl_name() << endl;
		cout << "annual salary: " << increament() << endl;

	}
};

int main() {
	cout<<"ROLL NO.: 22K-4638  class: 2-f"<<endl;
	Bank_Employee b1("Mohammad", "irtiza", 50000), b2("Mohammad", "sadiq",
30000);
	cout << "actual salary: " << endl;
	b1.display();
	b2.display();
    cout<<endl;
	cout << "after giving raise in salary: " << endl;
	b1.DISPLAY();
	b2.DISPLAY();


	return 0;
}
```

OUTPUT:

```
ROLL NO.: 22K-4638  class: 2-f
actual salary:
f_name: Mohammad
l_name: irtiza
annual salary: 600000
f_name: Mohammad
l_name: sadiq
annual salary: 360000

after giving raise in salary:
f_name: Mohammad
l_name: irtiza
annual salary: 720000
f_name: Mohammad
l_name: sadiq
annual salary: 432000
```

QUESTION # 04

Write a grading program for a class with the following grading policies:
a. There are two quizzes, each graded on the basis of 10 points.
b. There is one midterm exam and one final exam, each graded on the basis of 100 points.
c. The final exam counts for 50% of the grade, the midterm counts for25%, and the two quizzes together count for a total of 25%. (Do not forget to normalize the quiz scores. They should be converted to a percent before they are averaged in.) Any grade of 90 or more is an A, any grade of 80 or more (but less than 90) is a B, any grade of 70 or more (but less than 80) is a C, any grade of 60 or more (but less than 70) is a D, and any grade below 60 is an F. The program will read in the student's scores and output the student's record, which consists of two quiz and two exam scores as well as the student's average numeric score for the entire course and the final letter grade. Define and use a structure for the student record.

CODE:

```
//ROLL NO.: 22K-4638  class: 2-f
#include<iostream>
#include<string>
using namespace std;
struct CLASS
{
    char grade;
    float Quiz[2];
    float quiz;
    float mid;
    float Final;
    float total;

}C1;

void calculate();
void get_marks(float a[2],float b,float c){
C1.Quiz[0]=a[0];
C1.Quiz[1]=a[1];
C1.mid=b;
C1.Final=c;

calculate();
}

void getGrade ();

void calculate( ){
    C1.quiz=C1.Quiz[0]+C1.Quiz[1];
    C1.total=(1.25*C1.quiz)+(0.25*C1.mid)+(0.5*C1.Final);
    getGrade();

}


void getGrade(){
    if(C1.total>=90){
        C1.grade='A';
    }
    else if(C1.total>=80&&C1.total<90){
        C1.grade='B';
```

```
    }
    else if(C1.total>=70&&C1.total<80){
        C1.grade='C';
    }
    else if(C1.total>=60&&C1.total<70){
        C1.grade='D';
    }
    else{
        C1.grade='F';
    }
}


void display(){
    cout<<"STUDENT RECORD IS: "<<endl;
    cout<<"QUIZ 1 SCORE: "<<C1.Quiz[0]<<endl;
    cout<<"QUIZ 2 SCORE: "<<C1.Quiz[1]<<endl;
    cout<<"MID SCORE: "<<C1.mid<<endl;
    cout<<"FINAL SCORE: "<<C1.Final<<endl;
    cout<<"TOTAL SCORE: "<<C1.total<<endl;
    cout<<"GRADE: "<<C1.grade<<endl;
}


int main(){
cout<<"ROLL NO.: 22K-4638  class: 2-f"<<endl;
float quiz[2],mid,Final;

cout<<"Enter the details: "<<endl;
cout<<"Enter the quiz 1 marks: "<<endl;
cin>>quiz[0];
cout<<"Enter the quiz 2 marks: "<<endl;
cin>>quiz[1];
cout<<"Enter the mid marks: "<<endl;
cin>>mid;
cout<<"Enter the final marks: "<<endl;
cin>>Final;
get_marks(quiz,mid,Final);
display();
return 0;
}
```

OUTPUT:

```
ROLL NO.: 22K-4638  class: 2-f
Enter the details:
Enter the quiz 1 marks:
10
Enter the quiz 2 marks:
10
Enter the mid marks:
75
Enter the final marks:
92
STUDENT RECORD IS:
QUIZ 1 SCORE: 10
QUIZ 2 SCORE: 10
MID SCORE: 75
FINAL SCORE: 92
TOTAL SCORE: 89.75
GRADE: B
```

QUESTION #05

You are a programmer for the ALFALAH Bank assigned to develop a class that models the basic workings of
a bank account. The class should perform the following tasks:

- Save the account balance.
- Save the number of transactions performed on the account.
- Allow deposits to be made to the account.
- Allow withdrawals to be taken from the account.
- Calculate interest for the period.
- Report the current account balance at any time.
- Report the current number of transactions at any time.

Menu

1. Display the account balance
2. Display the number of transactions
3. Display interest earned for this period
4. Make a deposit
5. Make a withdrawal
6. Add interest for this period
7. Exit the program

CODE:

```
//ROLL NO.: 22K-4638 class: 2-f
#include<iostream>
#include<string>
using namespace std;
class BankAccount{
float Balance;
static int transaction;
float interest_rate;
float Interest;
public:
BankAccount(){}
BankAccount(float initialBalance=0,float rate=0.01){
interest_rate= rate;
transaction=0;
// set_interestAmount(period);
if(initialBalance>=0){
Balance=initialBalance;
}
else{
Balance=0;
cout<<"initial balance was invalid"<<endl;
}
}

//get balance
float get_balance(){
return Balance;
}
```

```
//cedit
void credit(float amount ){
Balance+=amount;
transaction++;
}

//debit
void debit(float amount){
if(amount> Balance){
cout<<"debit amount exceed the account balance"<<endl;
}
else{
Balance-=amount;
transaction++;
}

}

//transaction
int get_transaction(){
return transaction;
}

//interest
void set_interestAmount(int period){
Interest=period*interest_rate*Balance;
Balance+=Interest;
}

float get_interestRate(){
return interest_rate;
}
};
int BankAccount :: transaction(0);

int main(){
cout<<"ROLL NO.: 22K-4638 class: 2-f"<<endl;
BankAccount account(1000);

int period;
float amount;
int choice;

do{
cout<<"Menu"<<endl;
cout << "1. Display the account balance" << endl;
cout << "2. Display the number of transactions" << endl;
cout << "3. Display interest earned for this period" << endl;
```

```
cout << "4. Make a deposit" << endl;
cout << "5. Make a withdrawal" << endl;
cout << "6. Add interest for this period" << endl;
cout << "7. Exit the program" << endl;
cout << "Enter your choice: ";
cin >> choice;
switch(choice){
case 1:
cout<<"Your account Balance is: "<<account.get_balance()<<endl;
break;
case 2:
cout<<"No. of transaction are: "<<account.get_transaction()<<endl;
break;
case 3:
cout<<"Enter the period in month: "<<endl;
cin>>period;
//account.set_interestAmount(period);
cout<<"Interest for the given period is:
"<<(account.get_interestRate()*period*account.get_balance())<<endl;
break;
case 4:
cout<<"Enter the amount for deposit: "<<endl;
cin>>amount;
account.credit(amount);
cout<<"Amount deposited successfully"<<endl;
break;
case 5:
cout<<"Enter the amount for withdraw: "<<endl;
cin>>amount;
account.debit(amount);
break;
case 6:
cout<<"Enter the period in month: "<<endl;
cin>>period;
account.set_interestAmount(period);
cout<<"Interest added to the balance"<<endl;
break;
case 7:
cout<<"terminating the program!"<<endl;
break;
default:
cout<<"invalid choice"<<endl;
}
}while(choice!=7);
}
```

OUTPUT:

```
ROLL NO.: 22K-4638  class: 2-f
Menu
1. Display the account balance
2. Display the number of transactions
3. Display interest earned for this period
4. Make a deposit
5. Make a withdrawal
6. Add interest for this period
7. Exit the program
Enter your choice: 1
Your account Balance is: 1000
Menu
1. Display the account balance
2. Display the number of transactions
3. Display interest earned for this period
4. Make a deposit
5. Make a withdrawal
6. Add interest for this period
7. Exit the program
Enter your choice: 2
No. of transaction are: 0
Menu
1. Display the account balance
2. Display the number of transactions
3. Display interest earned for this period
4. Make a deposit
5. Make a withdrawal
6. Add interest for this period
7. Exit the program
Enter your choice: 3
Enter the period in month:
5
Interest for the given period is: 50
```

```
Menu
1. Display the account balance
2. Display the number of transactions
3. Display interest earned for this period
4. Make a deposit
5. Make a withdrawal
6. Add interest for this period
7. Exit the program
Enter your choice: 4
Enter the amount for deposit:
3000
Amount deposited successfully
Menu
1. Display the account balance
2. Display the number of transactions
3. Display interest earned for this period
4. Make a deposit
5. Make a withdrawal
6. Add interest for this period
7. Exit the program
Enter your choice: 1
Your account Balance is: 4000
Menu
1. Display the account balance
2. Display the number of transactions
3. Display interest earned for this period
4. Make a deposit
5. Make a withdrawal
6. Add interest for this period
7. Exit the program
Enter your choice: 5
Enter the amount for withdraw:
2500
Menu
```

```
Menu
1. Display the account balance
2. Display the number of transactions
3. Display interest earned for this period
4. Make a deposit
5. Make a withdrawal
6. Add interest for this period
7. Exit the program
Enter your choice: 1
Your account Balance is: 1500
Menu
1. Display the account balance
2. Display the number of transactions
3. Display interest earned for this period
4. Make a deposit
5. Make a withdrawal
6. Add interest for this period
7. Exit the program
Enter your choice: 6
Enter the period in month:
5
Interest added to the balance
Menu
1. Display the account balance
2. Display the number of transactions
3. Display interest earned for this period
4. Make a deposit
5. Make a withdrawal
6. Add interest for this period
7. Exit the program
Enter your choice: 1
Your account Balance is: 1575
Menu
1. Display the account balance
```

```
Menu
1. Display the account balance
2. Display the number of transactions
3. Display interest earned for this period
4. Make a deposit
5. Make a withdrawal
6. Add interest for this period
7. Exit the program
Enter your choice: 7
terminating the program!
```