# Lab 03 (Dated 2-10-2020)

**1. Address**

Incomplete Address class and main function is given. Write member functions in Address class, used in main function. Write getter setter functions for all data members. Output is inside box in front of main function. Write code with all consideration of OOP concepts discussed so far like const, public & private.

```
class Address{
   int houseNo;//1-8
   char floor;//a-d
   char block;//A-J
};
int main(){
   int h;
   char f, b;
   cin>>h>>f>>b;
   Address a1(h, f, b);
   cin>>h>>f>>b;
   Address a2(h, f, b);
   a1.show();
   a2.show();
   if (a1.isSameBlock(a2))
         cout<<"has same block\n";
   else
         cout<<"has different block\n";
   cin>>f;
   a1.setFloor(f);
   if (a1.hasUpperFloor(a2))
         cout<<"has upper floor\n";
   else
         cout<<"has same or lower floor\n";
   a2.setFloor('c');
   cout<<"A2 has floor:"<<a2.getFloor()<<'\n';
   cout<<"Block:"<<a2.getBlock()<<'\n';
   cout<<"House no:"<<a1.getHouseNo()<<'\n';
   return 0;
}
```

**Input Format**

2 a C
4 b C
d

**Output Format**

House No:2
Floor:a
Block:C

---

House No:4
Floor:b
Block:C

---

has same block
has upper floor
A2 has floor:c
Block:D
House no:2

**Solution:**

```
class Address{
    int houseNo;//1-8
    char floor;//a-d
    char block;//A-J
public:
    Address(int h, char f, char b){
        set(h, f, b);
    }
    void set(int h, char f, char b){
        setHouse(h);setBlock(b);setFloor(f);
    }
    void setHouse(int houseNo){
        if (houseNo<=0 || houseNo>8) houseNo=1;
        this->houseNo=houseNo;
    }
    void setFloor(char floor){
        if (floor<'a' || floor>'d') floor='a';
        this->floor=floor;
    }
    void setBlock(char block){
        if (block<'A' || block>'J') block='a';
        this->block=block;
    }
    int getHouseNo() const{
        return houseNo;
    }
    char getFloor() const{
        return floor;
    }
    char getBlock() const{
        return block;
    }
    void show() const{
        string s="House No:";
        s+=(char)(houseNo%10+'0');
        s=s+"\nFloor:"+floor+"\nBlock:"+block+'\n';
        cout << s+"------------\n";
    }
    bool isSameBlock(const Address &a){
        return block==a.block;
    }
    bool hasUpperFloor(const Address &a){
        return floor>a.floor;
    }
};
int main(){
    int h; char f, b;
    cin>>h>>f>>b;
    Address a1(h, f, b);
    cin>>h>>f>>b;
```

```
    Address a2(h, f, b);
    a1.show();
    a2.show();
    if (a1.isSameBlock(a2))
        cout<<"has same block\n";
    else
        cout<<"has different block\n";
    cin>>f;
    a1.setFloor(f);
    if (a1.hasUpperFloor(a2))
        cout<<"has upper floor\n";
    else
        cout<<"has same or lower floor\n";
    a2.setFloor('c');
    cout<<"A2 has floor:"<<a2.getFloor()<<'\n';
    cout<<"Block:"<<a2.getBlock()<<'\n';
    cout<<"House no:"<<a1.getHouseNo()<<'\n';
    return 0;
}
```

**2. Class Employee**

Complete class Employee with appropriate data members and write definition of member functions declared. Employee class has four data members, see two constructor declarations. In setter of hours, check if hours are less than 20, assign 20 by default. Similarly, in setter of rate, check if rate is less than 50, assign 50 by default. Call setters from constructor. In function increase hour rate, simply add rate to existing rate. Details of function to calculate salary is as follows:

Multiply hours with hour rate, If hours are greater than 40 then add 50% extra payment for extra hours. See output for further understanding of functions.

```
class Employee{
      //data members
public:
      Employee(string fn, string sn);//assign 30 to hours and 100 hour rate
      Employee(string fn, string sn, int h, int hR);//called setters inside
      void setHours(int h);
      void setHourRate(int hR);
      void increaseHourRate(int inc);
      void show();
      int calcSalary();
};
int main(){
      string sN, fN;
      int h, hR;
      cin>>fN>>sN;
      Employee emp1 (sN, fN);
      cin>>fN>>sN;
      cin.clear();
      cin>>h>>hR;
      Employee emp2 (sN, fN, h, hR);
      emp1.show();
      emp2.show();
      cout << emp1.calcSalary() << ' ' << emp2.calcSalary() << '\n';
```

```
    return 0;
}
```

**Input Format**
Aslam Sajid
Farooq Qaiser 35 60
Constraints
No Constraint

**Output Format**
Aslam Sajid 30 100 3000
Farooq Qaiser 35 60 2100
3000 2100

**Solution:**

```
class Employee{
    string fName;
    string sName;
    int hours;
    int hourRate;
public:
    Employee(string fn, string sn){
        fName=fn;
        sName=sn;
        hours=30;
        hourRate=100;
    }
    Employee(string fn, string sn, int h, int hR){
        fName=fn;
        sName=sn;
        setHours(h);
        setHourRate(hR);
    }
    void setHours(int h){
        if (h<20)    h=20;
        hours=h;
    }
    void setHourRate(int hR){
        if (hR<50)    hR=50;
        hourRate=hR;
    }
    void increaseHourRate(int inc){
        hourRate+=inc;
    }
    void show(){
        cout << sName << ' ' << fName << ' ' << hours << ' ' << hourRate << ' '
<< calcSalary() <<'\n';
    }
    int calcSalary(){
        int sal=hours*hourRate;
        if (hours>40)
            sal+=(hours-40)*hourRate*0.5;
        return sal;
    }
};
int main(){
    string sN, fN;
    int h, hR;
```

```
    cin>>fN>>sN;
    Employee emp1 (sN, fN);
    cin>>fN>>sN;
    cin.clear();
    cin>>h>>hR;
    Employee emp2 (sN, fN, h, hR);
    emp1.show();
    emp2.show();
    cout << emp1.calcSalary() << ' ' << emp2.calcSalary() << '\n';
    return 0;
}
```

**3. Matrix Operations**

A 2D array has else In this problem, matrix operations are required. There can be matrix of any size. You have to read rows and columns. Next read matrix elements accordingly. Next read a positive number, representing number of operations. There may be any number of operations possible, however there are 7 different possible operations, which may be called in any order:

1. multiply kth row with some scalar value
2. multiply kth col with some scalar value
3. add jth row into kth row
4. add jth col into kth col
5. subtract jth row from kth row
6. subtract jth col from kth col
7. show matrix elements in proper format

After reading number of operations. User has to read required operation number. For each operation further read operation may be required according to the operation. User has to perform operation according to operation number. See input. There are two rows three columns. Next there are 6 elements. There are 5 operations. There are three show operations represented by 7. Whereas two other operations. See output for further understanding.

**Input Format**

```
2 3
3 1 5
4 2 6
5
7
2
2 4
7
6
1 2
7
```

Output Format

```
3 1 5
4 2 6
3 4 5
4 8 6
3 1 5
4 4 6
```

**Solution:**

```
class Matrix{
    int **d;
    int rows, cols;
public:
```

```
    Matrix(int r, int c){
        rows = r;
        cols = c;
        d=new int*[rows];
        for (int i=0;i<rows;i++)
            d[i]=new int[cols];
    }
    void read(){
        int i,j;
        for (i=0;i<rows;i++)
            for (j=0;j<cols;j++)
                cin>>d[i][j];
    }
    void show() const{
        int i,j;
        for (i=0;i<rows;i++){
            for (j=0;j<cols;j++)
                cout << d[i][j] << ' ';
            cout << '\n';
        }
    }
    void multiplyKthRow(int k, int v){
        for (int i=0;i<cols;i++)
            d[k-1][i] *= v;
    }
    void multiplyKthCol(int k, int v){
        for (int i=0;i<rows;i++)
            d[i][k-1] *= v;
    }
    void addJthColIntoKthCol(int j, int k){
        for (int i=0;i<rows;i++)
            d[i][k-1] += d[i][j-1];
    }
    void subtractJthColFromKthCol(int j, int k){
        for (int i=0;i<rows;i++)
            d[i][k-1] -= d[i][j-1];
    }
    void addJthRowIntoKthRow(int j, int k){
        for (int i=0;i<cols;i++)
            d[k-1][i] += d[j-1][i];
    }
    void subtractJthRowFromKthRow(int j, int k){
        for (int i=0;i<cols;i++)
            d[k-1][i] -= d[j-1][i];
    }
    ~Matrix(){
        for (int i=0;i<rows;i++)
            delete []d[i];
        delete []d;
    }
};
int main(){
```

```
    int r, c, j, k, v, n, option;
    cin>>r>>c;
    Matrix m(r, c);
    m.read();
    cin>>n;
    for (int i=0;i<n;i++){
        cin>>option;
        switch(option){
            case 1://multiply kth row
                cin>>k>>v;
                m.multiplyKthRow(k, v);
                break;
            case 2://multiply kth col
                cin>>k>>v;
                m.multiplyKthCol(k, v);
                break;
            case 3://add jth row into kth row
                cin>>j>>k;
                m.addJthRowIntoKthRow(j, k);
                break;
            case 4://add jth col into kth col
                cin>>j>>k;
                m.addJthColIntoKthCol(j, k);
                break;
            case 5://subtract jth row from kth row
                cin>>j>>k;
                m.subtractJthRowFromKthRow(j, k);
                break;
            case 6://subtract jth col from kth col
                cin>>j>>k;
                m.subtractJthColFromKthCol(j, k);
                break;
            case 7://show()
                m.show();
        }
    }
    return 0;
}
```