C++ Online A1

Write a C++ class "Matrix" encapsulating a two-dimensional integer matrix object. The class should have the following member functions

1. A **constructor** which takes two integer arguments (number of row and number of column) and is responsible for necessary memory allocation (you should use malloc () and free () function for memory allocation).
2. A **destructor** with the responsibility of freeing memory
3. A **print** function with the responsibility of displaying matrix
4. A **set** function which takes three integer arguments (row, column, value) with the responsibility of setting the values of a matrix element.
5. A **get** function which takes two integer arguments (row, column) with the responsibility of returning the values of a matrix element.
6. An **add** function which takes an integer argument (n) which is added with each element of the matrix encapsulated by the object accessing add function
7. Another overloaded **add** function which add all the element of the matrix encapsulated by the object accessing add function and return the sum

You can use the following main function

```
int main()
{
   cout<<"Hello World"<<'\n';
   Matrix m(3,3);
   for(int i=0; i<3; i++)
      for(int j=0; j<3; j++)
         m.set(i,j,i+j);
   m.print();
   cout<<m.get(0,0)<<'\n';
   m.set(0,0,100);
   cout<<m.get(0,0)<<'\n';
   m.add(100);
   m.print();
   cout<<m.add()<<'\n';
   return 0;
}
```

Your output should be like the following:

```
Hello World
0 1 2
1 2 3
2 3 4
0
100
200 101 102
101 102 103
102 103 104
1018
```

C++ Online A2

Write a C++ class "Vectors" encapsulating a one-dimensional array of 3D position vectors. The class should have the following member functions

1. A **constructor** which takes an integer argument (maximum number of vector in the array) and is responsible for necessary memory allocation (you should use malloc() and free() function for memory allocation).
2. A **destructor** with the responsibility of freeing memory
3. A **print** function with the responsibility of displaying all the vectors (one vector per row)
4. A **set** function which takes four integer arguments (index, x, y, z) with the responsibility of setting the values of an array element.
5. A **get** function which takes an integer argument (index) with the responsibility of returning the coordinates of a vector as an integer array (you can use integer pointer).
6. An **add** function which takes a vector using three integer arguments (x, y, z) that is added with each vector of the array encapsulated by the object accessing add function
7. Another overloaded **add** function which add all the vectors of the array encapsulated by the object accessing add function and return the resultant vector as an integer array (you can use integer pointer).

You can use the following main function

```
int main()
{
    cout<<"Hello World"<<'\n';

    Vectors m(3);
    for(int i=0; i<3; i++)
        for(int j=0; j<3; j++)
            m.set(i,i+j,i-j,i*j);

    m.print();
    int* array = m.get(0);
    cout<<array[0]<<' '<<array[1]<<' '<<array[2]<<'\n';
    m.set(0,100,100,100);
    array = m.get(0);
    cout<<array[0]<<' '<<array[1]<<' '<<array[2]<<'\n';
    m.add(100,100,100);
    m.print();

    array = m.add();
    cout<<array[0]<<' '<<array[1]<<' '<<array[2]<<'\n';

    return 0;

}
```

Your output should be like the following:

```
Hello World
2 -2 0
3 -1 2
4 0 4
2 -2 0
100 100 100
200 200 200
103 99 102
104 100 104
407 399 406
```

C++ Online B1

Write a C++ class “Matrix” encapsulating a two-dimensional integer matrix objects. The class should have the following member functions

1. A **constructor** which takes two integer arguments (number of row and number of column) and is responsible for necessary memory allocation (you should use malloc() and free() function for memory allocation).
2. A **destructor** with the responsibility of freeing memory
3. A **print** function with the responsibility of displaying matrix
4. A **set** function which takes three integer arguments (row, column, value) with the responsibility of setting the values of a matrix element.
5. A **get** function which takes two integer arguments (row, column) with the responsibility of returning the values of a matrix element.
6. An **add** function which takes an integer argument (n) which is added with each element of the matrix encapsulated by the object accessing add function
7. Another overloaded **add** function which add all the element of the matrix encapsulated by the object accessing add function and return the sum

You can use the following main function

```
int main()
{
    cout<<"Hello World"<<'\n';

    Matrix m(3,3);
    for(int i=0; i<3; i++)
        for(int j=0; j<3; j++)
            m.set(i,j,i+j);

    m.print();
    cout<<m.get(0,0)<<'\n';
    m.set(0,0,100);
    cout<<m.get(0,0)<<'\n';
    m.add(100);
    m.print();
    cout<<m.add()<<'\n';

    return 0;
}
```

Your output should be like the following:

```
Hello World
0 1 2
1 2 3
2 3 4
0
100
200 101 102
101 102 103
102 103 104
1018
```

C++ Online B2

Write a C++ class “Random” encapsulating a two-dimensional random integer matrix objects. The class should have the following member functions

1. A **constructor** which takes four integer arguments (number of row, number of column, minimum, maximum) and is responsible for necessary memory allocation (you should use malloc() and free() function for memory allocation).
2. A **destructor** with the responsibility of freeing memory
3. A **print** function with the responsibility of displaying matrix
4. A **randomize** function with the responsibility of setting the values of a matrix element with random number (you can use rand() function).
5. A **get** function which takes two integer arguments (row, column) with the responsibility of returning the values of a matrix element.
6. An **add** function which takes an integer argument (n) which is added with each element of the matrix encapsulated by the object accessing add function
7. Another overloaded **add** function which add all the element of the matrix encapsulated by the object accessing add function and return the sum

You can use the following main function

```
int main()
{
    cout<<"Hello World"<<'\n';

    Matrix m(3,3,100,200);
    m.randomize();
    m.print();
    cout<<m.get(0,0)<<'\n';
    m.add(100);
    m.print();
    cout<<m.add()<<'\n';

    return 0;
}
```

Your output should be like the following:

```
Hello World
183 186 177
115 193 135
186 192 149
183
283 286 277
215 293 235
286 292 249
2416
```