Name - DHRUV DIXIT
Class - BCA
Roll no -: 2310997073

M T W T F S S
Page No.:
Date:
YOUVA

Assignment - 2

4.)
```
#include <iostream>
using namespace std;

class demo {
private:
  int A[rows][cols];

public:
  void get_data(){
   for(int i = 0; i<rows; i++){
    for( int j = 0; j< cols ; j++){
      cin >> A[i][j];
    }
   }
  }

  void show_data(){
   for(int i = 0; i< rows; i++){
    for(int j=0; j< cols; j++){
     cout << A[i][j] << "\t";
    }
    cout << endl;
   }
  }

  demo operator*(const demo &obj) const {
    demo result;
    for(int i = 0; i< rows; i++){
     for(int j=0; j< cols; j++){
       result.A[i][j] = 0;
        for(int k=0; k< cols; k++){
```

```
        result.A[i][j] += A[i][k] * obj.A[k][j];
    }
  }
  return result;
 }
};

int main() {
  demo d1, d2, d3;
  cout << "Enter data for first matrix:" << endl;
  d1.get_data();
  cout << "Enter data for second matrix:" << endl;
  d2.get_data();

  cout << "First matrix:" << endl;
  d1.show_data();
  cout << "Second matrix:" << endl;
  d2.show_data();

  d3 = d1 * d2;
  cout << "Product of matrices:" << endl;
  d3.show_data();

  return 0;
}
```

2)
```
# include <iostream>
# include <string>
using namespace std;

class student {
protected:
    string name;
    int age;

public:
    student (const string & name, int age): name
    (name), age (age) {}

    void display () {
    cout << "Name :" << name << endl;
    cout << " Age :" << age << endl;

    }
};

class Transport : public student {
private:
    string transportType;

public:
    Transport (const string & name, int age,
    const string & transportType)
        : student (name, age), transportType
            (transportType) {}

    void displayTransportDetails() {
        display();
```

```
    cout << "Transport Type: " << transportType <<
                                        endl;
  }
};

int main() {

  Transport student1("John", 20, "Bus");

  Transport class
    cout << "student Details :" << endl;
    student1. displayTransportDetails();

    return 0;

}
```

Date: YOUVA

3.) #include <iostream>
using namespace std;

class Test {
private:
int data1, data2, data3;

public:
Test(int d1=0, int d2=0, int d3=0):
data1(d1), data2(d2), data3(d3){}

void get_data(int d1){
data1 = d1;
}

void get_data(int d1, int d2){
data1 = d1;
data2 = d2;
}

void get_data(int d1, int d2, int d3){
data1 = d1;
data2 = d2;
data3 = d3;
}

void show_data(){
cout << "Data1: " << data1 << endl;
cout << "Data2: " << data2 << endl;
cout << "Data3: " << data3 << endl;
cout << endl;
}

3:

```
int main(){

Test obj1, obj2, obj3;

obj1.get_data(10);
obj2.get_data(20,30);
obj3.get_data(40,50,60);
cout << " Obj 1:" << endl;
cout << " Obj 2:" << endl;
cout << " Obj 3:" << endl;

return 0;
```

3

4.)
```
#include <iostream>
using namespace std;

class BCA_OOPS{
private:
  int var1;
  int var2;

public:

  BCA_OOPS(): var1(0), var2(0){}

  BCA_OOPS(int v1, int v2): var1(v1), var2(v2),
                            ~~var3(v3)~~ .{}

  void assignValues(int v1, int 2){
  this -> var1 = v1;
  this -> var2 = v2;
}

friend void displayValues(const BCA_OOPS & obj);

void copyData(const BCA_OOPS & obj){
  this -> var1 = obj.var1;
  this -> var2 = obj.var2;
}

void decrementValues(){
  if (var1 >= 0 && var1 <= 10)
    var1 -= 2;
  else if (var1 > 10)
    var1 -= 5;
```

```
if (var2 >= 0 && var2 <= 10)
    var2 = 2;
else if (var2 > 10)
    var2 -= 5;
}
};

void displayValues (const BCA_OOPS & obj) {
    cout << "var1: " << obj.var1 << endl;
    cout << "var2: " << obj.var2 << endl;
}

int main()

    BCA_OOPS obj1;

    cout << "Values of obj1 (Default constructor): " << endl;
    displayValues(obj1);

    BCA_OOPS obj2(15,8);

    cout << " Values of obj2 (Parameterized constructor): "
    << endl;
    displayValues( obj2);

    obj1.assignValues(5,12);

    cout << " Values of obj1 after using assign
    values() function: " << endl;
    displayValues(obj1);

    return 0;
}
```