DRY-RUN EXERCISES

9.1 What will happen when the following program is run?

```
#include <iostream>
using namespace std;
class B
{
 float b;
 public:
 B()
 { b = 5.0; }
 void show()
 {
  cout << b << endl;
 }
};

class D : public B
{
 float d;
 float d1;
 public:
 D()
 {
  B::B();
  d = 10.0;
  d1 = 20.0;
 }
 void show()
 {
  cout << d << endl;
 }
};
int main()
{
 B ob, *ptr;
 D ob1;
 ptr = &ob1;
 ptr->show();
 cout << ptr << endl;
 ptr++;
 cout << ptr;
 return 0;
}
```

.2 What will happen when the following program is run?

```
#include <iostream>
using namespace std;
int main()
{
     int ar[6]={22,5,23,43,54,65};
     int *p,i;
     p=ar;
     i=*p++ - *p--;
     cout<<i;
     return 0;
}
```

.3 What will happen when the following program is run?

```
#include <iostream>
using namespace std;
void square(int * snum)
{
    cout <<  "Square of 10 is ";
    *snum *= *snum;
}
int main()
{
    int num = 10;
    square(&num);
    cout << num << endl;
    return 0;
}
```

Scanned with CamScanner

9.4 What will be the output of the following program?

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

class p
{
public:
    virtual void print()
    {
        cout<<" it is base class \n";
    }
};
class q: public p
{
public:
    void print()
    {
        cout<<"it is Derived class \n";
    }
};
int main()
{
    p *b = new q;
    b->print();
    return 0;
}
```

9.5 What will be the change in output if virtual keyword is removed from print function of class original_base?

```
#include<iostream>
using namespace std;
class original_base
{
public:
    virtual void print ()
    { cout<< "print version of base class" <<endl; }
    void show ()
    { cout<< "show version of base class" <<endl; }
};

class derived_from :public original_base
{
public:
    void print ()
    { cout<< "print version of derived class" <<endl; }
    void show ()
    { cout<< "show version of derived class" <<endl; }
};

int main()
```

```
{
    original_base *ptr;
    derived_from x;
    ptr = &x;
    ptr->print();
    ptr->show();
}
```

9.6 What will happen when the following program is run?

```
#include <iostream>
using namespace std;
class B
{
 float b;
 public:
 B()
 { b = 5.0; }
 virtual void show()
 {
  cout << b << endl;
 }
};

class D : public B
{
 float d;
 float d1;
 public:
 D()
 {
  B::B();
  d = 10.0;
  d1 = 20.0;
 }
 void show()
 {
  cout << d << endl;
 }
};

int main()
{
 B ob, *ptr;
 D ob1;
 ptr = &ob;
 cout << sizeof(*ptr) << endl;
 ptr = &ob1;
 cout << sizeof(*ptr) << endl;
 return 0;
}
```

9.7 What will happen when the following program is run?

```
#include <iostream>
using namespace std;
class B
{
 float b;
 public:
 B()
 { b = 5.0; }
 void show()
 {
  cout << b << endl;
 }
};

class D : public B
{
 float d;
 float d1;
 public:
 D()
 {
  B::B();
  d = 10.0;
  d1 = 20.0;
 }
 void show()
 {
  cout << d << endl;
 }
};

int main()
{
 B ob, *ptr;
 D ob1;
 ptr = &ob;
 cout << sizeof(*ptr) << endl;
 ptr = &ob1;
 cout << sizeof(*ptr) << endl;
 return 0;
}
```

9.8 What will happen when the following program is run?

```
#include <iostream>
using namespace std;
class B
{
 float b;
 public:
```

```
B()
{ b = 5.0; }
void show()
{
 cout << b << endl;
}
};
class D : public virtual B
{
 float d;
 float d1;
 public:
 D()
 {
  B::B();
  d = 10.0;
  d1 = 20.0;
 }
 void show()
 {
  cout << d << endl;
 }
 };

 int main()
 {
  B ob, *ptr;
  D ob1;
  ptr = &ob;
  cout << sizeof(*ptr) << endl;
  ptr = &ob1;
  cout << sizeof(*ptr) << endl;
  return 0;
 }
```