Date:.....

ITEM 11:

Handle assignment to self in operator =

Suppose we have a class employee which has many data members like name, salary & other info,

```
Class Employee { };

Employee E1, E2;

E1 = E2; // calls Operator =
E2 = E2; // calls operator = (self assignment)
// done by mistake.
```

Now, operator = will be called and all the info of Employee will be copied to its own object, to avoid this overhead we can match the address of this & the parameter coming, if both are same or not.

```
class Bitmap { };

class Widget {
    private
        Bitmap *pb;
};
```

Date:.....

```
Widget & Widget :: operator = (const Widget &rhs)
{
    if (this == &rhs)  // match address
        return *this;      of this & rhs

    delete pb;
    pb = new Bitmap(*rhs.pb);
    return *this;
}
```

Since we are matching address, we are not using *this, *this points to value but this points address.

If we are deleting pb, this can be problem since if something like new Bitmap yields exception then Widget will hold pointer to a deleted Bitmap.

To avoid this, do not delete pb until we have copied what it points to :—

Date:.....

```
Widget& Widget::operator=(Widget &rhs)
{
    Bitmap *porig = pb;
    pb = new Bitmap(*rhs.pb);
    delete porig;
    return *this;
}
```

Even if some except comes in Bitmap's constructor due to memory full, pb we still have refernce of pb in porig.

=> We can achieve by copy & swap also.

```
class Widget {
    void swap(Widget &rhs);
};
Widget& Widget::operator=(Widget &rhs)
{   Widget temp(rhs); // Copy rhs data
    swap(temp); // swap *this with temp.
    return *this;
}
```