Date:.....

ITEM 7

Declare destructors Virtual

We have

```
Class Base {

};
Class Derived1 : public Base { };
Class Derived2 : public Base { };

Base *b = getBValue();
//get dynamically allocated object from
  Base heirarchy.

delete b; //release to avoid resource
            leak.
```

→ We cannot rely on client to free memory from Heap.

→ Also C++ specifies that when a derived class object is deleted thru a pointer to a base class with a non-virtual destructor, results are undefined.

→ derived part of object is never destroyed.

Date:.....

If base class has virtual destructor it will destroy its derived parts.

=> Declare a virtual destructor in a class if & only if that class contains at least one virtual functions.

Sine an int ~~Se~~ occupies 32 bit. Can fit into a 64 bit register.

but if destructor is virtual of the same class having only int member data, this inf takes form of pointer called vptr. vptr points to an array of function pointers called ~~Vtable~~ Vtable.

Thus on a 32 bit architecture, they will go from 64 bits (for 2 ints one member data, other vptr) to 96 bits. (for ints + vptr), thus taking lot of space. Thus, declare a virtual destructor if there is a virtual funcn.

Date:.....

Declare pure virtual destructor in class you want to be abstract.

Eg:

```
class A{
public:
    virtual ~A()=0;
};
```

Must provide definition for pure virtual destructor since it is called by its derived class destructor. If not done, ~~compiler~~ with linker will complain.

```
A::~A(){}
```