Date/...../.....

Item20: Prefer pass by Reference to const to pass by value.

Consider following example.

```
Class Person {
 public:
      Person();
      virtual ~Person();
      ...
 private:
      string name;
      string address;
};

class Student: public person {
 public:
      Student();
     ~Student();
      ...
 private:
      string schoolName;
      string schoolAddress;
};
```

Date:.....

now consider a functn

bool validateStudent(Student s);

Student s1;
bool ok = validateStudent (s1);

=> Student copy ctor called to initialize parameter s from s1.

=> s is destroyed when validateStudent returns

The order of events are.

=> Person ctor called.
=> Student copy ctor
=> student has two string objects, ctor of 2 string objects
=> ctor of two string of person also
=> dtor of 4 string object of Person & Student
=> dtor of ~~Person~~ Student
=> dtor of Person.

6 ctor & 6 destructor called if we pass by value.

Date:

To bypass all ctor & dtor pass by reference to const

bool validateStudent (const Student &s);

⇒ No ctor & dtor are called
⇒ const is used otherwise caller would have to worry about validateStudent making changes to Student object passed.

In pass by value, another copy of object is created so there is no fear of copy / constantness but in pass by reference better to use const.

⇒ Also if derived class object is passed by value to base class object object slicing will happen.

For example ⇒

```
Class Window
{ public:
    string name() const;
    virtual void display() const;
};
```

Date:

```
Class WindowWithScrollBars {
    public:
    virtual void display() const;
};

void printNameAndDisplay(Window W)
{ cout << w.name();
  w.display();          // object slicing
}
```

If we call:

```
WindowsWithScrollBars wwsb;
printNameAndDisplay(wwsb);
```

The call to display inside printNameAndDisplay will always call Window::Display.

hence wwsb will be sliced off.

=> All this bcoz references are implemented as pointers.

```
void printNameAndDisplay(const Window &W)
                    // won't be sliced.
```

Date:.....

if we have object of built in type, then passing by value is better than pass by reference. since it is small.
also STL iterators & function object types.