```
2021-10-10 -- Scratchpad of CSE213 (Sec-1)
==========================================

operator overloading (cont.):
=============================

Q: What happens when we write cin>>x; (where x is an int) ?
A: operator method "istream& operator>>(int&);" of istream class is
executed

Q: What happens when we write cin>>str; (where str is a string) ?
A:
        - the expected operator method "istream& operator>>(string&);"
does
          not exist in istream class, as string is a class

        - If we could add a new operator method "istream& operator>>
(string&);"
          to istream class, our problem could be solved. But we are
not allowed
          to modify a library class

        - as a second choice, string class author could not add
          "istream& operator>>(istream&);" to string class, because in
that case
          the client MUST be string object and istream object will be
parameter.
          IN THAT CASE, the call would look like: "str>>cin;". Since
the call
          "str>>cin" is not acceptable, then this is not a method of
string class
          either.

        - In reality, string class author add "istream& operator>>
(istream&,string&);"
          this as a global function in the cpp file which contains
string class

--------------------------------------
if we now want to write:
"cin>>c1"; instead of "c1.setComplexNo();"

we also need to add "istream& operator>>(istream&,ComplexNo&);"
his as a global function in the cpp file which contains ComplexNo
class
(refer cpp file)

class OneDArray{        //refer image in recording
        int sz;
        int* valPtr;
        public:
        int operator[](int index){
                return valPtr[index];
        }
}
----------
```

```
OneDArray arr1;
//code to new & populate with data
cout<<arr1.vaPtr[2]; //if fields are public
cout<<arr1.getVaPtrAt(2); //if fields are pritave
cout<<arr1[2];

------------------------------------------
int main(){
        ComplexNo c1,c2;
        cout<<"Enter real & img for c1: "; cin>>c1;     //1 2
        cout<<"c1 = "<<c1<<endl;                        //1+2i
        //++c1;
        c2 = ++c1;
        //cout<<"after ++C1, C1 = "<<c1<<endl;
        cout<<"after c2= ++c1, c1 = "<<c1<<endl;        //2+3i
        cout<<"after c2= ++c1, c2 = "<<c2<<endl;        //
        return 0;
}

int i=10, j=10, k;

cout<<++i;      //i will be 11, and 11 will be used for output
cout<<j++;      //current value of j (10) will be used for output,
then j will be 11
cout<<"i="<<i<<", j="<<j<<endl;


++i;
i++;
these two statements are same when they are used alone
But their impact in expression are not same, if (++) is used in an
expression

when used in expression (++i),
incremented value will be used in expression

when used in expression (i++),
current value will be used in expression, then it will be incremented

=============================================================================
=====
Customer Requirement Analysis:
-------------------------------
As a  deliverable of milestone-1, you need to produce a document
called
CRA-report. CRA report contains:

--------------------------------- for IRAS ---------------------------
-
User-1: Student
        Goal-1: Register course for upcoming semester
                Workflow:
                  e-1: check for being defaulter
(doc/payment/provision)
                  e-2: date/time/slot varification
                  e-3: load applicable courses for add
                  .....
                  e-n: ..........

        Goal-2: Evaaluate faculty for a registered course of current
semester
                Workflow:
                  e-1: ......
                  e-2: ......
                  e-3: ......
                  .....
                  e-n: ..........

        Goal-3: ....
        Goal-4: .....
        Goal-5: .......


User-2: Faculty
        Goal-1: .....
        Goal-2: ......
        Goal-3: ....
        Goal-4: .....
        Goal-5: .......

...

User-n: xxxxxx
        Goal-1: .....
        Goal-2: ......
        Goal-3: ....
        Goal-4: .....
        Goal-5: .......

=====================================================
```