Problem 1:

```
vector<int> v1;
vector<int> v2;
if (v1 = v2)
    :
```

Template...

```
class mVector
{public:
bool operator==(mVector<T> v2);
}
```

Template...

```
bool mVector<T>::operator==(mVector<T> v2)
{ if (vsize != v2.size())
      return false;
  for (int i=0; i<v.size; i++)
      { if (v[i] != v2[i])
          { return false;
          }
      }
  return true;
}
```

Problem 2:

* Wrote Problem on board *

```
{ int sum = 0;
  for (int i=0; i<n; i++)   | n times
  {
    for (int j=0; j<n*n; j++)
      { sum++; }
  }
  return sum;
}
```

i	iterations
0	n^2
.	n^2
.	
.	
.	
.	
n-1	n^2

a. $T(n) = 2n^3 + n$

b. $O(n^3)$

$$\lim_{n\to\infty} \frac{2n^3+n}{n^3} = 2$$

or

$T(n) = 2n^3 + n \leq 2n^3 + n^3 = 3n^3$

for $n \geq 1$

```
LNode
  { T data;
    LNode <T>*lptr;
    LNode <T>* Rptr;
```

3:

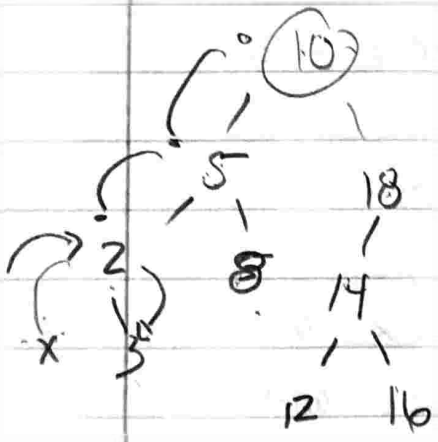
```
class mList
 { Public:
   void erase () { erase (first); }


Privat:
   LNode <T> * First
   LNode <T> * Last
    int lsize;
  void erase (Lnode <T> * PTR);
    }

void mlist <T>:: erase (LNode <T> * Ptr)
    {
        if (ptr==0)
            { return 0;
              }

      else
          { erase (ptr -> rPtr);
              delete ptr;
                lsize--;
                 }

}
```

Problem 4

```
template...
void mTree<T>::delete(T x)
public { delete( root , T x);

}

Void mTree<T>::delete(TNode<T> * ptr, T x)
    if ( ptr->lptr->data == x and
         ptr->lptr->lptr == 0 and
         ptr->lptr->rptr == 0)

        delete ptr->lptr;
        ptr->lptr = 0;
    if ( ptr->rptr->data == x and
         ptr->rptr->lptr == 0 and
         ptr->rptr->rptr == 0)
                    {
        delete ptr->rptr;
        ptr->rptr = 0;
                    }

    return
```

```
if (xc = ptr -> data)
    { delete (ptr -> lptr, x);
else {
    delete (ptr -> rptr, x);

return
}
```

Prob 5



10, 5, 2, 3, 8
18, 14, 12, 16