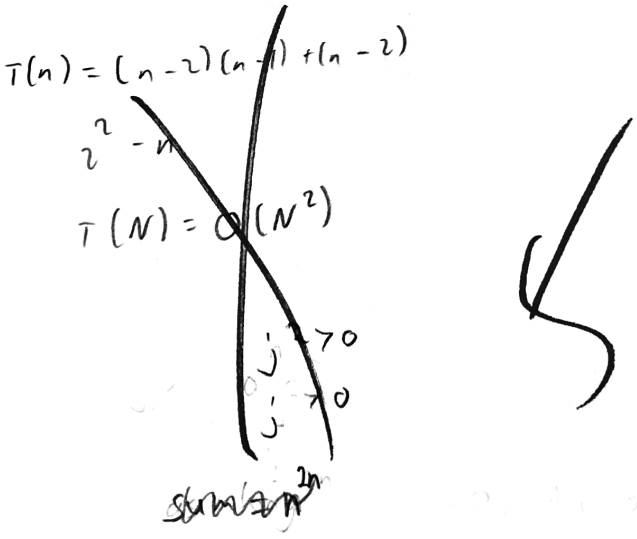
Problem 1. (20 pts) Consider the following program segment below.

```
sum=0;
for (i = 0; i < n*n; i++)
    {
    for (j = n; j > 0; j--)
        {
           sum++;
        }
    }
```

(a) Compute the time complexity function of the program segment as a function of the value n.

(b) Classify the function from Part a using the O notation.



Problem 2. (20 pts) Assume the Mlist class does not have the integer *lsize* attribute which stores the length of the Mlist. Write a new function *size* which returns the length of the list by counting the number of nodes in the list.

```
int lsize;
template <typename T>
void Mlist<T>::size()
{
    Lnode<T> *next = ptr->ptr;
    Lnode<T> *previous = ptr->lptr;
    if (ptr > 0)
    {
        lsize++;
    }
    else
    {
        return lsize;
    }
}
```

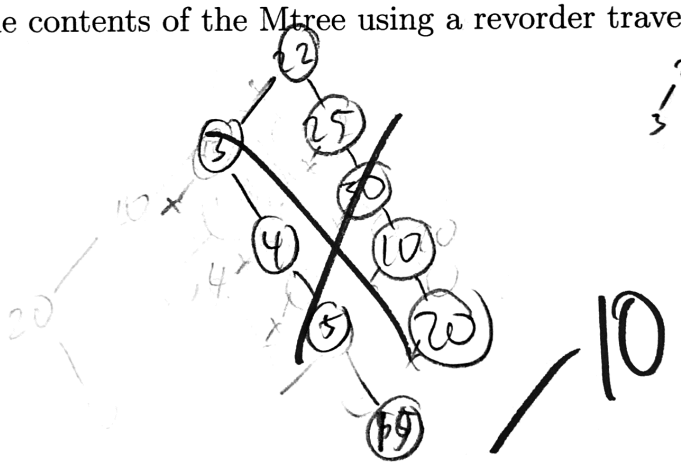
Problem 3. (20 pts) Consider the *revorder* (reverse order) traversal of a tree given below :

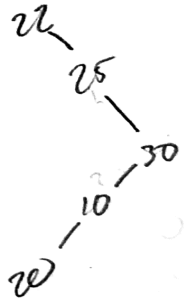
```
Visit right tree
Visit root
Visit left tree
```

(a) Build an Mtree (using the $\leq$ comparison) for the values added in the order 20 10 30 25 22 15 5 4 3.

(b) List the contents of the Mtree using a revorder traversal.

a)





10

b) 20, 10, 30, 25, 22, 15, 5, 4, 3

Problem 4. (20 points) Write the revorder function for the class Mtree.

```
void mtree<T>::revorder(Tnode<T> *ptr)
{
  if(ptr==0)
  {
    return;
  }
  else
  {
    revorder(ptr->rptr);
    v.push_back(ptr->data);
    revorder(ptr->lptr);
  }
}
```

Problem 5. (20 pts total) Write the erase functions for the Mtree class that deletes every node in the Mtree. Note there should be two erase functions. The first is in the *public* area and the second is a recursive function in the *private* section that is called by the function in the *public* area. (Hint : Use a traversal - Left tree - Right Tree - Root.)

```
template <typname T>
 void Mtree<T>::erase(x)
 {
   delete (root,x);
 }

template <Typname T>
 void Mtree<T>::erase (Tnode<T> *ptr)
 {
   if (ptr == root and ptr -> data == x and ptr -> lptr == 0 and rptr == 0)
    {
      delete ptr;
      size --;
      root = 0;
      return;
    }
 }
```