Exercise 1

A[n]
B[m]
C[n+m]
D[n+m]

Assuming we follow the rules of sets where there are no duplicates in the original sets

```
Copy contents of A and B into D.
Sort(D)
for(int j = 0; j < n+m-2; j++;)
        if D[j] == D[j+1]
                j += 2
        else
                Add D[j] to C
                j++
if(D[n+m-1] != D[n+m-2])
        Add D[n+m-1] to C

Return C
```

Exercise 2

```
bool Heap(int heapAr[],  int n)
{
   for (int i=0; i<=(n-2)/2; i++)
   {
      if (heapAr[2*i +1] > heapAr[i])
            return false;
      if (2*i+2 < n && heapAr[2*i+2] > heapAr[i])
            return false;
   }
   return true;
}
```

This algorithm visits the root and every internal node at each node checks that neither child is larger than itself with an efficiency of O(n)

Exercise 3

Heap sort is not stable because it does not preserve the original order of equal elements. For example say you have the original list 1,5,2,3,2,6 and just to keep track of the twos lets call the first one 2_x and the second 2_y. So again the original list is 1,5,2_x, 3, 2_y, 6 after constructing the heap the array representation will be 1, 2_y, 2_x, 3, 5, 6. As you can see the order of the twos were flipped in construction and is therefore unstable.

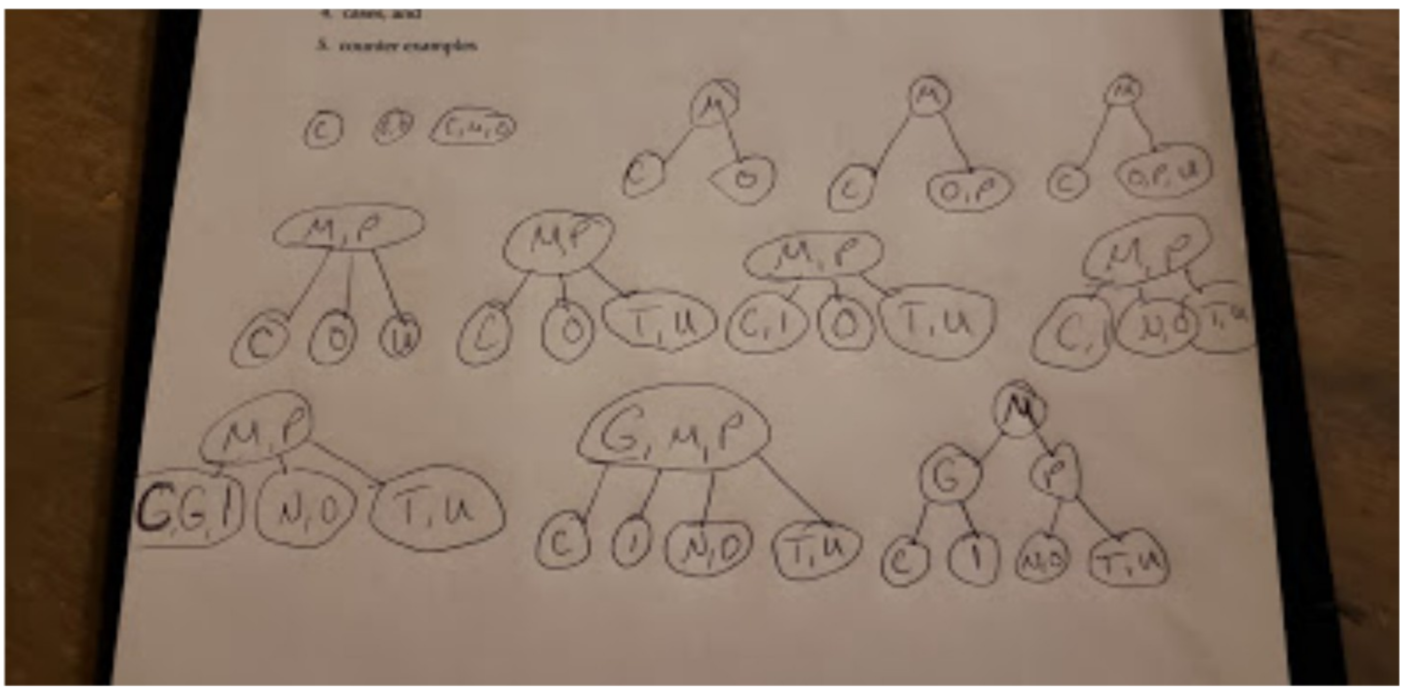
Exercise 4

z = .25(x+y)
x+y+z = 100
100 – z = x+y
z = .25(100-z)
z=25-.25z
z+.25z = 25
1.25z = 25
z = 20

x+y+z(20) = 100
x+y = 80
x = 1/3y
1/3y + y = 80
1.3333y = 80
y=60

x+y = 80
80-60 = x
x = 20

<u>Exercise 5</u>



The ones with most comparisons will be O and U being on the leafs as the last elements with 4 comparisons needed. For average time the probability 1/9 by the number of each required comparisons to get the average case result of 2.777777 comparisons.

<u>Exercise 6</u>

$p(x)= x^4-4x^3+7x^2-5x+2$ at $x=3$.

Coefficients	1	-4	7	-5	2
x=3	1	3*1+(-4)=-1	3*-1+7=4	3*4+(-5)= 7	3*7+2=23

b. The quotient and the remainder of the division of $x^4-4x^3+7x^2-5x+2$ by x-3 are $x^3 - x^2 + 4x + 7$ remainder: 23.