hw problems for June 4

```
```

The function Subsets(n, k) creates all subsets of range(1, n+1) of length k

However, to list them you should use Subsets(n,k).list() or list(Subsets(n,k))

```
Subsets(4,2).list()
```

```
[{1, 2}, {1, 3}, {1, 4}, {2, 3}, {2, 4}, {3, 4}]
```

The following function is one half of the answer to Exercise 1, in order to compete the problem you should write the function "subset_to_monomial" which takes a subset $SS = \{a_1 < a_2 < \ldots < a_{n-1}\}$ and converts it into the monomial

$$x_1^{a_1-1} x_2^{a_2-a_1-1} \cdots x_{n-1}^{a_{n-1}-a_{n-2}-1} x_n^{n+d-1-a_{n-1}}$$

```
def list_all_monomials(n, d):
    return [subset_to_monomial(SS, d) for SS in Subsets(n+d-1,d)]
```

```
list_all_monomials(3,2)
```

```
Traceback (click to the left of this block for traceback)
...
NameError: global name 'subset_to_monomial' is not defined
```

The following command make a list of 10 variables x0 through x9

```
vx = [var("x"+str(i)) for i in range(10)]
```

```
vx[3]
```

```
x3
```

The following is an expample of a product over a set

syntax:

mul(expr for var in aset)

```
mul( vx[i]^i for i in range(10) )
```

```
x1*x2^2*x3^3*x4^4*x5^5*x6^6*x7^7*x8^8*x9^9
```

```
```

You can also create a monomial with words instead of subsets, but since they are in bijection, they should be equivalent.

```
def num_0_eq_2( w ):
    return w.count('0')==2 # true if the number of 0's is equal to 2
```

```
Words(alphabet=['0','1'], length=4).filter( num_0_eq_2 ).list()
```

```
    [word: 0011, word: 0101, word: 0110, word: 1001, word: 1010, word:
    1100]
```

```
for w in Words(alphabet=[0,1], length=4):
    if count_zeros(w)==4:
        print w
```

```
    0000
```

```
w = Word(['0','1','0','1'])
```

```
LL = [[1,2],[3,4],[5,6,7]]
```

```
LL[0]
```

```
    [1, 2]
```

```
LL[0][0]
```

```
    1
```

```
[mul(var("x"+str(v)) for v in L if v>3) for L in LL]
```

```
    [1, x4, x5*x6*x7]
```

```
SS = [3,5,7]
```

```
deg = 7
```

```
SSp = [0]+SS+[deg+len(SS)+1]
```

```
mul(var("x"+str(j))^(SSp[j+1]-SSp[j]-1) for j in range(len(SSp)-1))
```

```
    x0^2*x1*x2*x3^3
```

```
def subset_to_monomial( SS, n ):
```

```
[0]+[1,2,3]+[8]
```

```
    [0, 1, 2, 3, 8]
```

```
w.count('0')
```

```
    2
```

```
w = Word([1,0,0,0,1,0,0,0,0,1])
```

```
for i in range(len(w)):
    print w[i]
```

```
1
0
0
0
1
0
0
0
0
1
```

```
out = 0
for i in range(len(w)):
    if w[i]==0:
        out=out+1
out
```

```
7
```

```
def count_zeros( w ):
    out = 0
    for i in range(len(w)):
        if w[i]==0:
            out=out+1
    return out
```

```
count_zeros(w)
```

```
7
```

```
w.count(0)
```

```
7
```

```
```