

In Python:

Problem #1

```
# Problem 1: taking a list of numbers and squaring each value
# the new values are then added to a new list
print()
print("Problem 1")
list1 = [1, 2, 3, 4, 5, 6, 7]
list2 = []
[list2.append(math.pow(x, 3)) for x in list1]
print (list2)
print()
```

Output:

```
Problem 1
[1.0, 8.0, 27.0, 64.0, 125.0, 216.0, 343.0]
```

Problem #2

```
# Problem 2: write a list comprehension that applies the
# fv function to items in the form of tuples
print("Problem 2")
list3 = [(1,3,5), (2,4,6)]
currelem = [fv(x,y,z) for x,y,z in list3]
print (currelem)
print()
```

Output:

```
Problem 2
[1024, 31250]
```

```
# The function fv() calculates the future value of an amount
# of money (principal) that is earning interest at a given
# rate over a given period of time.
def fv(principal, rate, time):

    curr = principal * ((1 + rate) ** time)
    return curr
```

Problem #3

```
# Problem 3: write a function abbreviator() that takes an input
# string and only keeps the upper-case characters from
# the original string, creating the abbreviation
print("Problem 3")
str = "How Are You"
newstr=""
print(str)
newlist = abbreviator(str)

for x in newlist:
    newstr += x

print(newstr)
print()
```

Output:

```
Problem 3
How Are You
HAY
```

```
# a function abbreviator() that uses list comprehension that
# takes an input string and only keeps the upper-case characters
# from the original string, creating the abbreviation.
def abbreviator(str):
    strlist=[]

    [strlist.append(x) for x in str if x.isupper()]

    return strlist
```

In Haskell:

Problem #4

```
[Prelude> [x**3 | x <- [1..5]]  
[1.0,8.0,27.0,64.0,125.0]
```

Problem #5

```
[Prelude> [y | y <-['A'..'Z'], y `elem` "HelloWorld"]  
"HW"
```