Problem 1:

**forgetting len: -1, forgetting set in a,c -1.**

```python
unic = len([x for x in set(nuts) if x[0] == 'c'])
solo = len([x for x in set(nuts) if nuts.count(x) == 1])
truenut = [x for x in set(nuts) if x.endswith("nut")]
nuttiest = sorted([(nuts.count(x),x) for x in nuts])[-1][1]
```

**Part E:**

```python
--
temp = sorted([(nuts.count(x),x) for x in set(nuts)])
freqs = [x[1] for x in temp]
```

### Problem 2, Part A

**-------------------

```python
def purchases(source):
    d = {}
    for line in source:
        line = line.strip().split(',')
        name = line[0]
        price = float(line[3])
        if name not in d:
            d[name] = 0
        d[name] += price
    data = sorted(d.items(),key=operator.itemgetter(1),reverse=True)
    for d in data:
        print "%s $%.2f" % (d[0],d[1])
```

---

Rubric is +5 process file/store for sorting/print
+3 sorting and printing
+5 process: read, split, use id, store total price
+5 this is to be able to sort, print, dictionary not only choice
+1: split to get at parts of line
+1: use indexes 1 and 3 (or 1 and -1) to identify parts of line
+1: call float somewhere to make a price that can be added
+2: store info, e.g., in dictionary and do that right e.g., for dictionary expect keys to be done right +1 and values +1
---

+3 sort and print
+2 call sort in a way that *could* work, i.e., identify item for sorting (typically this means reverse and sort-by-price,+1,+1)
+1 print properly
---

**Part B

-------------------

```python
def topsong(songd):
    d = {}
    for id in songd:
        songs = songd[id]
        for song in songs:
            d[song] = d.get(song,0) + 1
    data = sorted(d.items(), key=operator.itemgetter(1),reverse=True)
    return sorted[0][0]
```

---

Rubric is +5 +3. The +5 is before mx=0, the +3 is after
+5 loop and accumulate data
+3 find largest
+5 for loop/accumulate
+2 loop over each song
+2 accumulate values for finding top +1 details
+3 for return/max
+2 sort or find max
+1 return value
---

Alternate +5 +3. The +5 is before mx=0, the +3 is after
```python
def topsong(songd):
    songs = []
    for x in songd.values():
        song.extend(x)
    mx = 0
    ts = **
    for song in songs:
        if songs.count(song) > mx:
            mx = songs.count(song)
            ts = song
    return ts
```

---

**Problem 3

-------------------

```python
def getWinner(events):
    most = max([events.count(x) for x in events])
    d = {}
    for elt in events:
        d[elt] = d.get(elt,0) + 1
        if d[elt] == max:
            return elt
    return -1  # not reached
```

---

Rubric is +5 +3. The +5 is before mx=0, the +3 is after
+5 loop and accumulate data
+3 find largest
+5 for loop/accumulate
+2 loop over each song
+2 accumulate values for finding top +1 details
+3 for return/max
+2 sort or find max
+1 return value
---

--

**Part B

-------------------

```python
def getSortedList(kings):
    pairs = [x.split() for x in kings]
    data = [(x[0],x[1],roman_to_int(x[1])) for x in pairs]
    temp = sorted(data,key=operator.itemgetter(2))
    tups = sorted(temp,key=operator.itemgetter(0))
```

---

Rubric is +5 +3. The +5 is before mx=0, the +3 is after
+5 loop and accumulate data
+3 find largest
+5 for loop/accumulate
+2 loop over each song
+2 accumulate values for finding top +1 details
+3 for return/max
+2 sort or find max
+1 return value
---

Alternate +5 +3. The +5 is before mx=0, the +3 is after
```python
#tups = sorted(temp,key=operator.itemgetter(0,2)) # will work as one line

def pair_count(words):
    c = 0
    for i in range(len(words)):
        for j in range(i+1,len(words)):
            if iso(words[i],words[j]):
                c += 1
    return c
```

Problem 4:

---

Part A

- yes, yes, yes, no, reuse on b->y and d->y
- no, conflict on c->z and then c->y

Part B:

- addresses one of conflict and reuse
- does this right for one +3
- does this right for both +2
- tried to do both +1

Part C:

- Alternatively: reasonable start, but flawed in approach +2

- can lose points for extraneous code