```python
import argparse
import numpy as np
import os
NUM_MAX = 100
ITEM_MAX = 100

def generate_prob():
    price_max = NUM_MAX
    item_max = ITEM_MAX
    P = round(np.random.sample()* 1000, 2)
    M = round(np.random.sample()* 1000, 2)
    N = np.random.randint(0, 100)
    C = np.random.randint(0, item_max)
    num_class = N/2
    item_list = generate_item(P, M, N, num_class)
    constr_list = set(map(lambda x: str(x)[5:-2],
generate_constraint(C, num_class)))
    constr_list = map(lambda x: map(int, x.split(', ')), constr_list)
    C = len(constr_list)
    return (P, M, N, C, item_list, constr_list)

def generate_item(P, M, N, num_class):
    item_list = []
    for i in range(1, N+1):
        item_name = item_string(i)
        item_class = np.random.randint(num_class)
        item_weight = round(np.sqrt(P) + np.random.beta(1,3)*P, 2)
        item_cost = round(np.sqrt(M) + np.random.beta(0.5, 0.5) * M, 2)
        item_resale_val = round(np.random.beta(5, 1)* 2 * M, 2)
        item = (item_name, item_class, item_weight, item_cost,
item_resale_val)
        item_list.append(item)
    return item_list

def item_string(N):
    index = N
    string = ""
    base = ord('a')
    while index > 0:
        string = chr(base + (index-1)%26) + string
        index /= 26
    return string

def generate_constraint(C, num_class):
    lst = []
    for i in range(C):
        sizeOfClass = 2 + int(np.random.beta(2, 5) * (num_class-2))
        lst.append(set(np.random.randint(0, num_class, sizeOfClass)))
    return lst

def write_prob(prob, filename):
    try:
        P, M, N, C, items, constr = prob
        with open(filename, "w") as f:
            f.write('{}\n{}\n{}\n{}\n'.format(P, M, N, C))
            for item in items:
                f.write("{}; {}; {}; {}; {}\n".format(*item))
            for cons in constr:
                f.write(str(cons)[1:-1] + '\n')
        return True
    except:
        return False

if __name__ == "__main__":
    # parser = argparse.ArgumentParser(description="PickItems solver.")
    # parser.add_argument("file_name", type=str, help="____.in")
    # args = parser.parse_args()
    generated = 0
    while generated < 5:
        prob = generate_prob()
        result = write_prob(prob,
"simple_prob/problem{}.in".format(generated))
        file_size =
os.path.getsize("simple_prob/problem{}.in".format(generated)) >> 10
        print "file size: {}KB".format(file_size)
        # if 3500 <= file_size < 4000:
        generated += 1

    import ipdb; ipdb.set_trace()
```