```
from __future__ import division
import argparse
import numpy as np

"""
================================================================================
========
  Please complete the following function.
  Greedy Approach, get each selected candidate item exactly once.
================================================================================
========
"""
SELECT_RANGE = 10

def solve(P, M, N, C, items, constraints):
    """
    Write your amazing algorithm here.

    Return: a list of strings, corresponding to item names.
    """
    shop_list, buget, weight = [], M ,0
    candidates = pre_process(items, N)
    while len(candidates) > 0:
        sample = select(candidates, SELECT_RANGE)
        if buget - sample[3] < 0 or weight + sample[2] > P:
            candidates.remove(sample)
        else:
            shop_list.append(sample)
            buget -= sample[3]
            weight += sample[2]
            print len(shop_list), len(candidates), buget, weight
            candidates = remain(candidates, constraints, sample)
    # verify(shop_list, constraints)
    selected = map(lambda x: x[0], shop_list)
    return selected

def pre_process(items, N):
    candidates = []
    for i in range(N):
        if items[i][2] == 0 or items[i][3] == 0:
            value = np.inf
            continue
        value = (items[i][4] - items[i][3])/items[i][2]
        candidates.append(tuple(list(items[i]) + [value]))
    return sorted(candidates, key=lambda x: -x[5])

def select(candidates, S):
```

```
    if len(candidates) < S:
        items = candidates
    items = candidates[:S]
    index = np.random.choice(range(len(items)))
    return items[index]

def remain(candidates, constraints, sample):
    ralevent = [constr for constr in constraints if sample[1] in
constr]
    candidates.remove(sample)
    remove_list = []
    for constr in ralevent:
        for item in candidates:
            if item[1] == sample[1]:
                continue
            if item[1] in constr and sample[1] in constr:
                remove_list.append(item)
    for rm in remove_list:
        if rm in candidates:
            candidates.remove(rm)
    return candidates

def verify(selected, constraints):
    for constr in constraints:
        count = 0
        for item in selected:
            if item[1] in constr:
                count += 1
        if count > 1:
            print("Constraint conflict")
            return
    print("correct")

"""
================================================================================
========
  No need to change any code below this line.
================================================================================
========
"""

def read_input(filename):
    """
    P: float
    M: float
    N: integer
    C: integer
    items: list of tuples
    constraints: list of sets
```

```
    """
    with open(filename) as f:
        P = float(f.readline())
        M = float(f.readline())
        N = int(f.readline())
        C = int(f.readline())
        items = []
        constraints = []
        for i in range(N):
            name, cls, weight, cost, val = f.readline().split(";")
            items.append((name, int(cls), float(weight), float(cost),
float(val)))
        for i in range(C):
            constraint = set(eval(f.readline()))
            constraints.append(constraint)
    return P, M, N, C, items, constraints

def write_output(filename, items_chosen):
    with open(filename, "w") as f:
        for i in items_chosen:
            f.write("{0}\n".format(i))

if __name__ == "__main__":
    # parser = argparse.ArgumentParser(description="PickItems solver.")
    # parser.add_argument("input_file", type=str, help="____.in")
    # parser.add_argument("output_file", type=str, help="____.out")
    # args = parser.parse_args()
    # input_file, output_file = args.input_file, args.output_file
    print "Loading Input Files"
    for fi in range(21):
        input_file, output_file =
'project_instances_extracredit/problem{}.in'.format(fi+1),
'EC/problem{}.out'.format(fi+1)
        P, M, N, C, items, constraints = read_input(input_file)
        print "Start Solving..."
        items_chosen = solve(P, M, N, C, items, constraints)
        print "Finished Solving, Write to file."
        write_output(output_file, items_chosen)
```