#include

using namespace std;

int max_value;

vector W;
vector V;

int memo[1050][35];
int taken[1050][35];

int knapsack(int i, int w) {
if (i < 0 || w <= 0)="" return="" 0;="" if="" (memo[i][w]="" !="-1)" memo[i][w];="" (w[i]="">w) return
memo[i][w] = knapsack(i - 1, w);

```
auto not_take = knapsack(i - 1, w);
auto     take = knapsack(i - 1, w - W[i]) + V[i];

if (take > not_take) {
    taken[i][w] = true;
    return memo[i][w] = take;
    }
else return memo[i][w] = not_take;
```

}

void reconstruct(int i, int w){
stack > itens;

```
do {
    if (taken[i][w]) {
        w -= W[i];
        itens.push(make_pair(V[i], W[i]));
    }
} while(i--);

while(!itens.empty()) {
    printf("V: %d W: %d\n", itens.top().first, itens.top().second);
printf("Index: %d Index: %d\n", itens.top().first, itens.top().second);
itens.pop();
}
```

}

int main(){
int casos, objetos, p, w, pessoas, ppessoa;
scanf("%d", &casos);
for(int i=0; i<casos; i++){
max_value = 0;
V.clear();
W.clear();

```
      scanf("%d", &objetos);
      while(objetos--){
          scanf("%d %d", &p, &w);
          V.push_back(p);
          W.push_back(w);
      }
      scanf("%d", &pessoas);
      while(pessoas--){
          memset(memo, -1, sizeof memo);
          memset(taken, false, sizeof taken);
          scanf("%d", &ppessoa);
          max_value += knapsack(V.size(), ppessoa);
    reconstruct(V.size(), ppessoa);
        }
      printf("%d\n", max_value);
  }

  return 0;
```

}