#2050成绩

mn\_lst = [int(x) for x in input().split()]  
m, n = mn\_lst[0], mn\_lst[1]  
def sieve\_of\_eratosthenes(n):  
 primes = [True] \* (n+1)  
 primes[0] = primes[1] = False  
 p = 2  
 while p \* p <= n:  
 if primes[p]:  
 for i in range(p \* 2, n+1, p):  
 primes[i] = False  
 p += 1  
 prime\_numbers = [x for x in range(n+1) if primes[x]]  
 return prime\_numbers  
is\_primes=sieve\_of\_eratosthenes(10000)  
primes=[False]\*10001  
for x in is\_primes:  
 primes[x]=True  
anslst=[]  
for \_ in range(m):  
 grades=[int(x) for x in input().split()]  
 countes\_classes=len(grades)  
 his\_grade=0  
 for x in grades:  
 if x\*\*0.5%1==0 and primes[int(x\*\*0.5)]:  
 his\_grade+=x  
 if his\_grade==0:  
 ans=0  
 else:  
 ans="%.2f"%float(his\_grade/countes\_classes)  
 anslst.append(ans)  
for x in range(len(anslst)):  
 print(anslst[x])



#t\_primes

n=int(input())  
num\_lst=[int(x) for x in input().split()]  
def sieve\_of\_eratosthenes(n):  
 primes = [True] \* (n+1)  
 primes[0] = primes[1] = False  
 p = 2  
 while p \* p <= n:  
 if primes[p]:  
 for i in range(p \* 2, n+1, p):  
 primes[i] = False  
 p += 1  
 prime\_numbers = [x for x in range(n+1) if primes[x]]  
 return prime\_numbers  
primes=sieve\_of\_eratosthenes(10\*\*6+1)  
isprimes=[False]\*(10\*\*6+1)  
for x in primes:  
 isprimes[x]=True  
def is\_prime(x):  
 if isprimes[x]:  
 return True  
 else:  
 return False  
def is\_t\_prime(x):  
 if x==1 or x==0 or x\*\*0.5%1!=0:  
 return False  
 else:  
 if is\_prime(int(x\*\*0.5)):  
 return True  
for i in range(n):  
 if is\_t\_prime(num\_lst[i]):  
 print("YES")  
 else:  
 print("NO")



#圣诞老人的礼物

nwlst=[int(x) for x in input().split()]  
n,w=nwlst[0],nwlst[1]  
value\_dct={}  
weight\_dct={}  
vpw\_dct={}  
for i in range(n):  
 vwlst=[int(x) for x in input().split()]  
 value\_dct[i]=vwlst[0]  
 weight\_dct[i]=vwlst[1]  
 vpw\_dct[i]=float(vwlst[0]/vwlst[1])  
vpw\_lst=sorted(vpw\_dct,key=lambda x:vpw\_dct[x],reverse=True)  
load=0  
value=0  
box=0  
while load<w and box<n:  
 up\_load\_w=int(min(w-load,weight\_dct[vpw\_lst[box]]))  
 up\_load\_v=float(value\_dct[vpw\_lst[box]]/weight\_dct[vpw\_lst[box]]\*up\_load\_w)  
 value+=up\_load\_v  
 load+=up\_load\_w  
 box+=1  
print("%.1f"%value)



#Fraction类

def gcd(m,n):  
 while m%n !=0:  
 oldm=m  
 oldn=n  
 m=oldn  
 n=oldm%oldn  
 return n  
class Fraction:  
 def \_\_init\_\_(self,top,bottom):  
 self.num=top  
 self.den=bottom  
 def \_\_str\_\_(self,):  
 return str(self.num)+"/"+str(self.den)  
 def show(self):  
 print(self.num,"/",self.den)  
 def \_\_add\_\_(self, other):  
 newnum=self.num\*other.den+self.den\*other.num  
 newden=self.den\*other.den  
 common=gcd(newnum,newden)  
 return Fraction(newnum//common,newden//common)  
lst=[int(x) for x in input().split()]  
print(Fraction(lst[0],lst[1])+Fraction(lst[2],lst[3]))



#打怪兽  
nCases=int(input())  
ans\_lst=[]  
for \_ in range(nCases):  
 nmblst=[int(x) for x in input().split()]  
 n,m,b=nmblst[0],nmblst[1],nmblst[2]  
 act\_dct={}  
 for \_ in range(n):  
 act\_lst=[int(x) for x in input().split()]  
 ti,xi=act\_lst[0],act\_lst[1]  
 if ti in act\_dct.keys():  
 act\_dct[ti].append(xi)  
 else:  
 act\_dct[ti]=[xi]  
 x\_lst=sorted(act\_dct.keys())  
 t=0  
 while t <len(x\_lst):  
 t\_i=x\_lst[t]  
 act\_dct[t\_i].sort(reverse=True)  
 max\_xi=0  
 for i in range(min(m,len(act\_dct[t\_i]))):  
 max\_xi+=act\_dct[t\_i][i]  
 if b-max\_xi<=0:  
 ans\_lst.append(t\_i)  
 break  
 else:  
 b-=max\_xi  
 t+=1  
 else:  
 ans\_lst.append("alive")  
for x in ans\_lst:  
 print(str(x))

