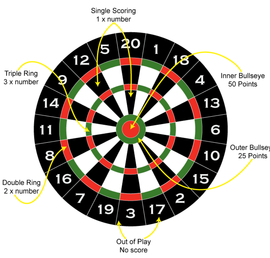
Darts 501

locked

Problem | Submissions | Leaderboard | Discussions



Submissions: 115
Max Score: 100
Difficulty: Medium

Rate This Challenge:

More

След успешния турнир по дартс на Freshers' Weekend 2019, от ФСС решиха да направят ежемесечен турнир на играта 501. С новия турнир обаче, дойдоха и нови тревоги за участниците. Те постоянно се чудеха какви и колко различни стратегии имат, които водят до бърза победа.

501 е най-популярната дартс игра, в която играчите започват с 501 точки и се стремят да стигнат до 0 с възможно най-малко на брой хвърляния, като се редуват да хвърлят по 3 стрели. Всички точки, които отбелязват, се изваждат от оставащите им точки.

Мишената е разделена на 20 части, отговарящи на числата от 1 до 20. Всяка от тези 20 части съдържа единичен сектор, който дава точки равни на неговата стойност, двоен сектор (дубъл), равняващ се на съответното число, умножено по 2 и троен сектор (требъл), който дава утроена стойност. В средата на мишената се намира центъра, който също е разделен на 2 части. Външният (по-голям) кръг дава 25 точки, а вътрешния (бул) е дубъл и дава 50. (Вижте картинката за по-подробна информация)

При едно хвърляне на 3 стрели максималния брой точки, който може да се постигне е 180 (3 x 60). За да се излезе и съответно да се спечели играта трябва да се направи така наречения „чекаут". "Чекаутът" е уцелване на точно толкова точки, колкото са ти останали. **Също така, за да се излезе е нужно последната стрела да уцели двоен сектор (дубъл) (вътрешният централен сектор също се брои за дубъл).**

Често срещан проблем пред дартс играчите е да разбрат дали и по колко начина мога да "изчистят" всичките си останали точки. От вас се иска да напишете програма, която изчислява по колко различни начина може да се направят това при оставащи **N** точки.

Имайте предвид, че е възможно играчът да пропусне мишената (да направи 0 точки). (вижте примерите)

Два чекаута са различни ако са от различен брой хвърляния (1, 2 или 3) или ако някое от хвърлянията се различава по уцелен сектор. (приемаме, че извън мишената е отделен сектор))

Повече информация за правилата на играта 501 можете да намерите тук:
http://www.nicedarts.com/how_to_501

Input Format

От първия ред се въвежда числото **N** – оставащите точки.

Constraints

$1 \leq N \leq 180$

Output Format

Отпечатайте едно число - по колко начина могат да се "изчистят" оставащите точки.

Sample Input 0

```
4
```

Sample Output 0

```
10
```

Explanation 0

Възможните излизания са:

С едно хвърляне: двойно 2

С две хвърляния: пропуск и двойно 2; 2 и двойно 1; двойно 1 и двойно 1

С три хвърляния: пропуск, пропуск и двойно 2; пропуск, 2 и двойно 1; пропуск, двойно 1 и двойно 1; 1, 1 и двойно 1; 2, пропуск и двойно 1; двойно 1, пропуск и двойно 1;

Следователно има общо 10 различни начина.

Sample Input 1

```
168
```

Sample Output 1

```
0
```

Explanation 1

Не е възможно точките да се изчистят с 3 хвърляния.

Sample Input 2

```
50
```

Sample Output 2

```
1371
```

Current Buffer (saved locally, editable) C++

```
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;
int isCheckoutPossible(int points){
    int counter=0;

    if(points>170||points<=0){
        return 0;
    }
    for(int k=0;k<=20;k++){

        for(int k=0;k<=20;k++){

            for(int l=1;l<=20;l++){

                if(k+k+l==points){
                    counter++;
                }
                if(k+2*k+l*2==points &&l!=0 ){
                    counter++;
                }
                if(l+k*2+l*2==points && k!=0){
                    counter++;
                }
                if(l+2*k+3*l==points && l!=0 && k!=0){
                    counter++;
                }
                 if(l+3*k+l*2==points && l!=0){
                    counter++;
                }
                 if(l+3*k+3*l*2==points && k!=0){
                    counter++;
                }
                if(l+3*k+3*l*2==points && l!=0 && k!=0 ){
                    counter++;
                }
                if(l+2*k+3*l*2==points && l!=0 && k!=0 ){
                    counter++;
                }
                if(l+3*k+2*l*2==points && l!=0 && k!=0 ){
                    counter++;
                }
                if(l+k+points-50  && l==20){
                    counter++;
                }
                if(l+2*k+points-50 && l!=0 && l==20){
                    counter++;
                }
                if(l+3*k+points-50 && l!=0 && l==20){
                    counter++;
                }
                if(l+2*k+2*points-50 && k!=0 && l==20){
                    counter++;
                }
                if(l+k+3*points-50 && k!=0 && l!=0 && l==20){
                    counter++;
                }
                if(l+2*k+2*points-50 && k!=0 && l!=0 && l==20){
                    counter++;
                }
                if(l+3*k+3*points-50 && k!=0 && l!=0 && l==20){
                    counter++;
                }
                if(l+3*k+2*points-50 && k!=0 && l!=0&& l==20){
                    counter++;
                }
                if(l+3*k+1*points-50 && k!=0 && l!=0 && l==20){
                    counter++;
                }

            }

        }
    }
    for(int k=1;k<=20;k++){

        for(int k=1;k<=20;k++){

            if(k+2*k+points){
                counter++;
            }
            if(k+2*k+2*points && l!=0){
                counter++;
            }
            if(k+2*k+3*points && l!=0){
                counter++;
            }
            if(l+points-50 && k==20){
                counter++;
            }
            if(l+2*points-50 && l!=0 && k==20){
                counter++;
            }
            if(l+3*points-50 && l!=0  && k==20){
                counter++;
            }
            if(l+25+k*2+points){
                counter++;
            }
            if(l+2*25+k*2+points && l!=0){
                counter+=2;
            }
            if(l+3*25+k*2+points && l!=0){
                counter+=2;
            }
             if(l+50+k*2+points){
                counter++;
            }
            if(l+2*50+k*2+points && l!=0){
                counter+=2;
            }
            if(l+3*50+k*2+points && l!=0){
                counter+=2;
            }
            if(l+25+50+points && k==20){
                counter++;
            }
            if(l+2*25+50+points && l!=0 && k==20){
                counter+=2;
            }
            if(l+3*25+50+points && l!=0 && k==20){
                counter+=2;
            }
            if(l+2*25*25+points && l!=0 && k==20){
                counter++;
            }
            if(l+2*50+50+points && l!=0 && k==20){
                counter++;
            }

            if(l+50+50+points && k==20){
                counter+=2;
            }
            if(l+2*50+50+points && l!=0 && k==20){
                counter+=2;
            }
            if(l+3*50+50+points && l!=0 && k==20){
                counter+=2;
            }
             if(k+2 + 25+points && l==20){
                counter++;
            }
             if(k+2 +50+points && l==20){
                counter+=2;
            }
            if(k+2 +25 +50+points && l==20){
                counter+=2;
            }

        }

    }
    if((points%2==0 && points<=40)||points==50){
        counter++;
    }
    if(points==100){
        counter++;
    }
    return counter;
}

int main() {
    int points;
    cin>>points;
    cout<<isCheckoutPossible(points);
    return 0;
}
```

Line: 1 Col: 1

Upload Code as File | Test against custom input | Run Code | Submit Code