OOP 期末上機考

A 卷

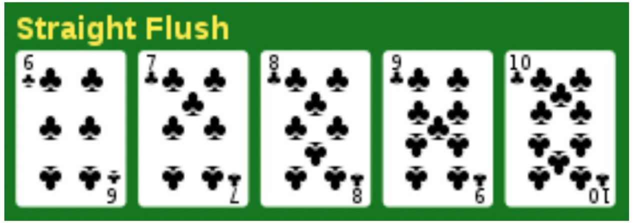
1.STL vector - **POKER (20%)**

Description

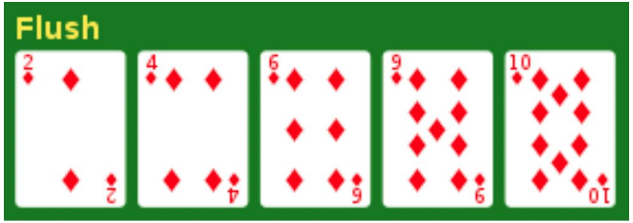
In poker, players construct hands of five cards according to predetermined rules, which vary according to which variant of poker is being played. A hand always consists of five cards. The suits of the cards are used to determine whether a hand forms a flush or straight flush.

In this case, there can be several categories of poker hands as follow pages.

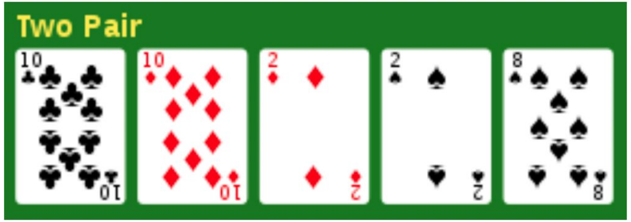
Straight flush



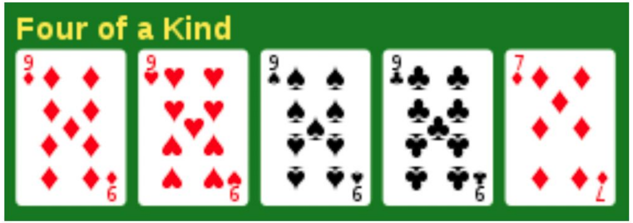
Flush



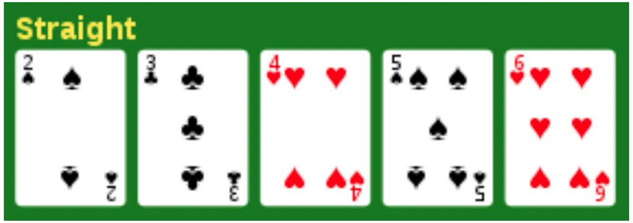
Two pair



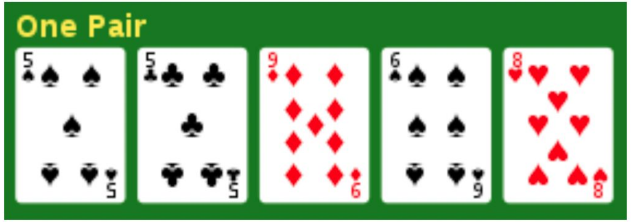
Four of a kind



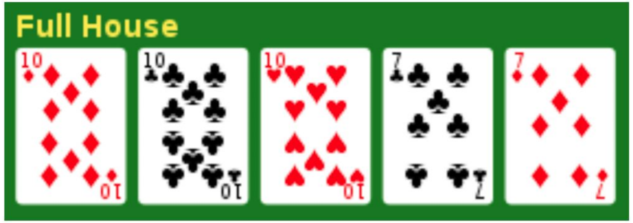
Straight



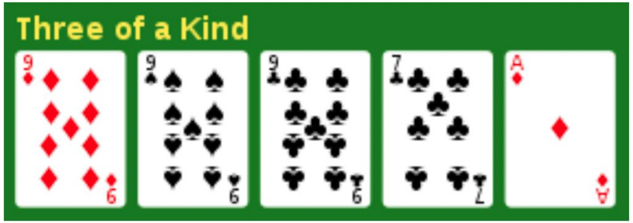
One pair



Full house



Three of a kind



Input

- The first line is the number of test cases. Each line of the test cases will contain five different strings separated by a single space to represent five cards. Each string consists two characters. The first one is **the number** of the card and the second one is **the suit** of the card.

- The card numbers are A,2,3,4,5,6,7,8,9,X,J,Q,K.Note that the numbers more than 9 are represented as X, J, Q, K, and 1 is represented as A. And the suits are S, H, D, C, all in capital letter, represent spade, heart, diamond and club. For your convenience, the five cards will be listed in **ascending sequence.**

- Input **ends** with a single row with the integer 0.

Output

- Input testcase: **input_pokerA.txt**
- Print the category of the poker hands and **a single empty row** between each group of test cases.

Sample Input	**Sample Output**
5 6C 7C 8C 9C XC 7D 9D 9H 9S 9C 7C 7D XD XC XH 2D 4D 6D 9D XD 2S 3C 4H 5S 6H 3 AD 7C 9D 9S 9C 2D 2S 8S XC XD 5S 5C 6S 8H 9D 0	Straight flush Four of a kind Full house Flush Straight Three of a kind Two pair One pair

2-STL Map **(20%)**

Given an array of unsigned integers and two other unsigned integers.

Example:

Input:

array = 2, 8, 57, 1, 0

x = 5

y = 3

Output: 4

Input:

array = 0, 1, 2

x = 1

y = 1

Output: 2

Description:

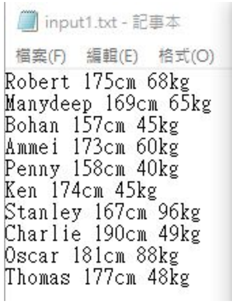
You have to convert the integers in the array into binary numbers. Find the size of the Largest subset of the array that there are at most 5 0’s and 3 1’s .

2 = 10, 8 = 1000, 57 =111001, 1 = 1, 0 = 0 , The subsets include {"1"}, {"0"}, {"10"}, {"1000"}, {"0", "1"}, {"0", "10"}, {"0", "1000"} , {"1", "10"}, {"1000", "1"} , {"1000", "10"} ,{"10", "1", "0"} ,{"1000", "1", "0"} ,{"10", "1000", "0"} ,{"10", "1", "1000"}and {"10", "1000", "1", "0"}.The maximum size of the subset is 4.

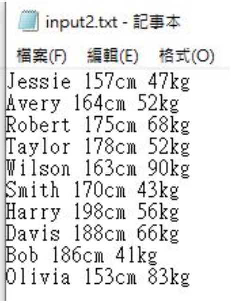
3-Classes **(25%)**

Define and implement a new class “student_data”

- These are 3 private data members
  - name
  - height
  - weight
- Record all student’s data and calculate their BMI and physical conditions. BMI = kilograms/(meters^2).
- Record height, weight and count BMI with data type double

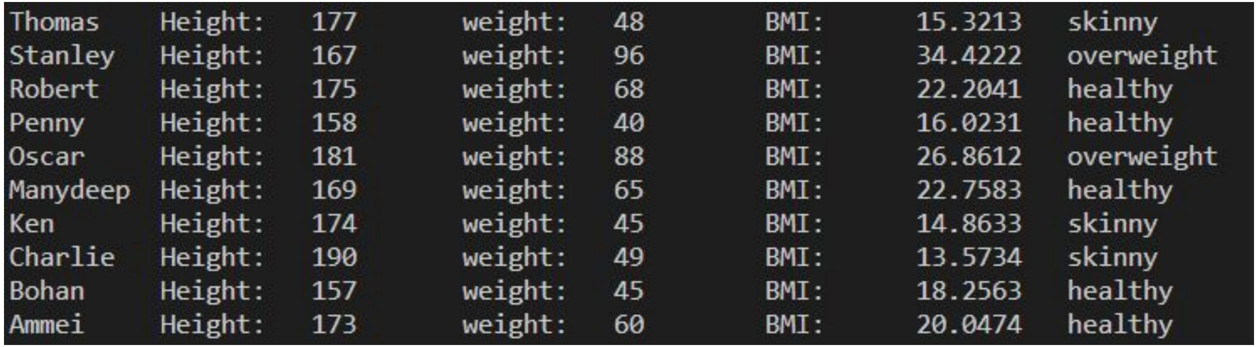


```
Robert 175cm 68kg
Manydeep 169cm 65kg
Bohan 157cm 45kg
Ammei 173cm 60kg
Penny 158cm 40kg
Ken 174cm 45kg
Stanley 167cm 96kg
Charlie 190cm 49kg
Oscar 181cm 88kg
Thomas 177cm 48kg
```



```
Jessie 157cm 47kg
Avery 164cm 52kg
Robert 175cm 68kg
Taylor 178cm 52kg
Wilson 163cm 90kg
Smith 170cm 43kg
Harry 198cm 56kg
Davis 188cm 66kg
Bob 186cm 41kg
Olivia 153cm 83kg
```

- Output: cout left and setw(10) ←→
  - student's information
    - Student's data
    - BMI
    - overweight (BMI>26) , skinny(BMI<16) or healthy
  - Sort student's name descending
  - Output example:

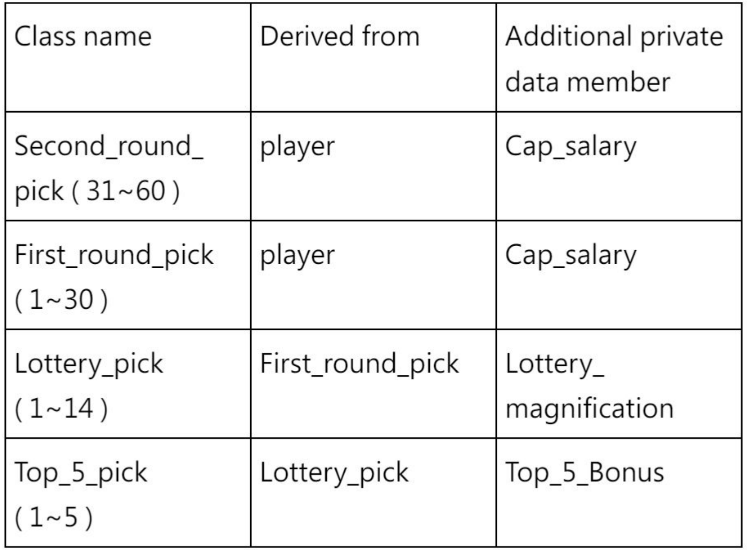


```
Thomas    Height:   177       weight:   48        BMI:      15.3213   skinny
Stanley   Height:   167       weight:   96        BMI:      34.4222   overweight
Robert    Height:   175       weight:   68        BMI:      22.2041   healthy
Penny     Height:   158       weight:   40        BMI:      16.0231   healthy
Oscar     Height:   181       weight:   88        BMI:      26.8612   overweight
Manydeep  Height:   169       weight:   65        BMI:      22.7583   healthy
Ken       Height:   174       weight:   45        BMI:      14.8633   skinny
Charlie   Height:   190       weight:   49        BMI:      13.5734   skinny
Bohan     Height:   157       weight:   45        BMI:      18.2563   healthy
Ammei     Height:   173       weight:   60        BMI:      20.0474   healthy
```

4-Inheritance

Assume that we have a brand-new calculating method for the salary of the NBA rookies, given the list of the draft result (Input.txt), output the draft pick、the name and the salary of the players in the output.txt.

With Input.txt, build up the following base and derived classes
Base class : player, with private data member (1) name (2) salary (3) pick



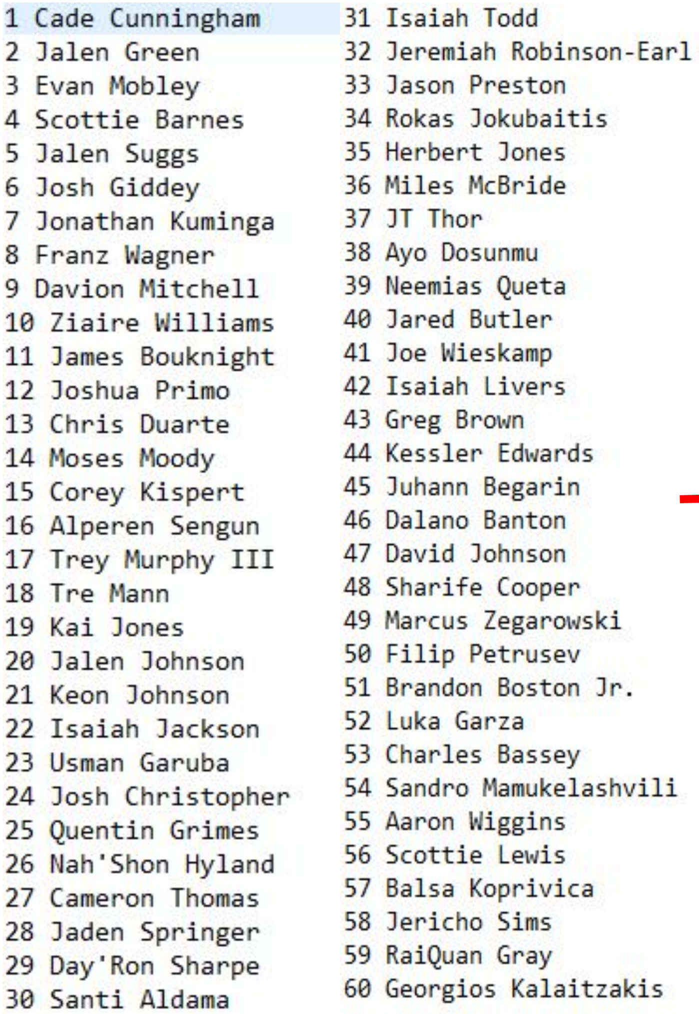
Class name	Derived from	Additional private data member
Second_round_ pick (31~60)	player	Cap_salary
First_round_pick (1~30)	player	Cap_salary
Lottery_pick (1~14)	First_round_pick	Lottery_ magnification
Top_5_pick (1~5)	Lottery_pick	Top_5_Bonus

4-Inheritance**(25%)**

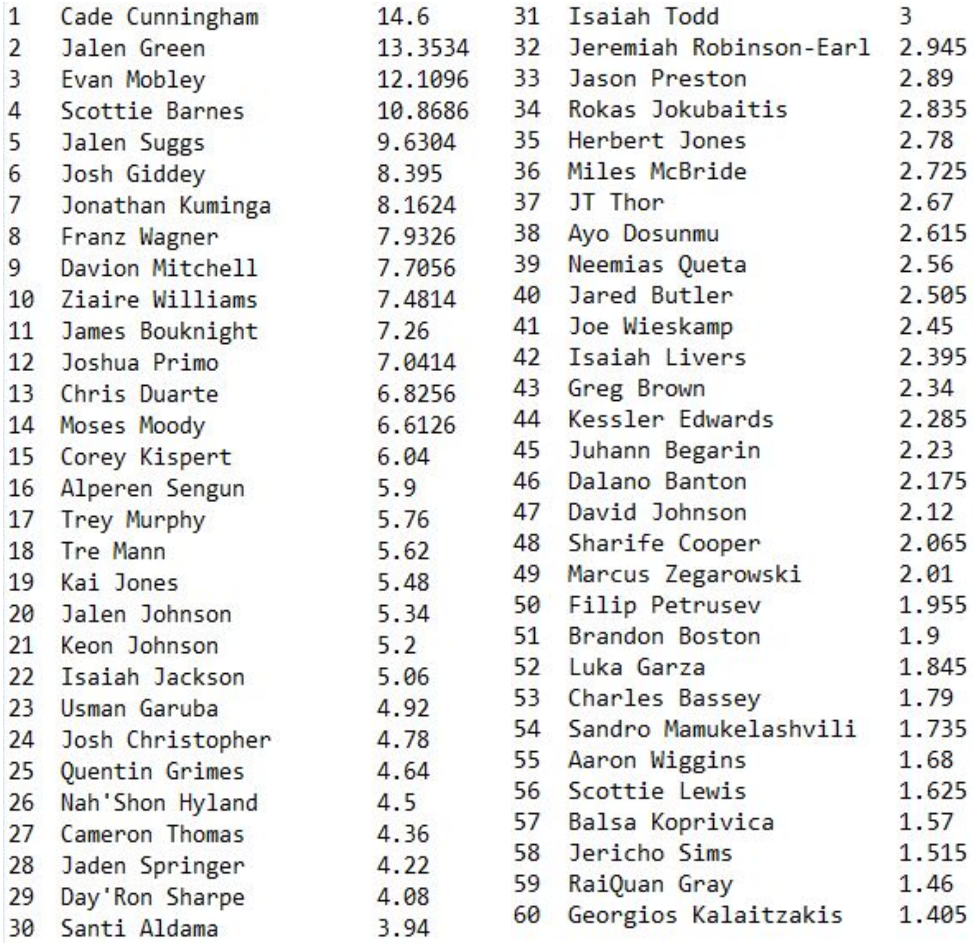
- Cap salary for 1~30 picks is 8M, and 3M for 31~60 picks.
- For 1~30 picks, 0.14M less salary for every going down of the pick.
- For 31~60 picks, 0.055M less salary for every going down of the pick.
- For the number one overall pick, his lottery magnification is 20%, and every one pick lower, 1% less of the lottery magnification (1~14 picks have lottery magnification).
- For the number one overall pick, his top-5 bonus is 5M, and every one pick lower, 1M less of the bonus (1~5 picks have top 5 bonus).
- Output format (all the information is left-aligned) : pick → setw(3) / name → setw(23)
- Final salary calculation :

  original salary * (1+lottery magnification) + top-5 bonus

pick	salary
1	8 x (1+0.2) + 5 = 14.6
6	7.3 x (1 + 0.15) = 8.395
15	6.04



1 Cade Cunningham
2 Jalen Green
3 Evan Mobley
4 Scottie Barnes
5 Jalen Suggs
6 Josh Giddey
7 Jonathan Kuminga
8 Franz Wagner
9 Davion Mitchell
10 Ziaire Williams
11 James Bouknight
12 Joshua Primo
13 Chris Duarte
14 Moses Moody
15 Corey Kispert
16 Alperen Sengun
17 Trey Murphy III
18 Tre Mann
19 Kai Jones
20 Jalen Johnson
21 Keon Johnson
22 Isaiah Jackson
23 Usman Garuba
24 Josh Christopher
25 Quentin Grimes
26 Nah'Shon Hyland
27 Cameron Thomas
28 Jaden Springer
29 Day'Ron Sharpe
30 Santi Aldama
31 Isaiah Todd
32 Jeremiah Robinson-Earl
33 Jason Preston
34 Rokas Jokubaitis
35 Herbert Jones
36 Miles McBride
37 JT Thor
38 Ayo Dosunmu
39 Neemias Queta
40 Jared Butler
41 Joe Wieskamp
42 Isaiah Livers
43 Greg Brown
44 Kessler Edwards
45 Juhann Begarin
46 Dalano Banton
47 David Johnson
48 Sharife Cooper
49 Marcus Zegarowski
50 Filip Petrusev
51 Brandon Boston Jr.
52 Luka Garza
53 Charles Bassey
54 Sandro Mamukelashvili
55 Aaron Wiggins
56 Scottie Lewis
57 Balsa Koprivica
58 Jericho Sims
59 RaiQuan Gray
60 Georgios Kalaitzakis

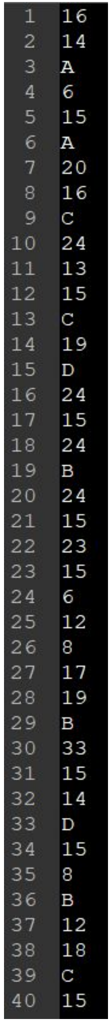


1	Cade Cunningham	14.6
2	Jalen Green	13.3534
3	Evan Mobley	12.1096
4	Scottie Barnes	10.8686
5	Jalen Suggs	9.6304
6	Josh Giddey	8.395
7	Jonathan Kuminga	8.1624
8	Franz Wagner	7.9326
9	Davion Mitchell	7.7056
10	Ziaire Williams	7.4814
11	James Bouknight	7.26
12	Joshua Primo	7.0414
13	Chris Duarte	6.8256
14	Moses Moody	6.6126
15	Corey Kispert	6.04
16	Alperen Sengun	5.9
17	Trey Murphy	5.76
18	Tre Mann	5.62
19	Kai Jones	5.48
20	Jalen Johnson	5.34
21	Keon Johnson	5.2
22	Isaiah Jackson	5.06
23	Usman Garuba	4.92
24	Josh Christopher	4.78
25	Quentin Grimes	4.64
26	Nah'Shon Hyland	4.5
27	Cameron Thomas	4.36
28	Jaden Springer	4.22
29	Day'Ron Sharpe	4.08
30	Santi Aldama	3.94
31	Isaiah Todd	3
32	Jeremiah Robinson-Earl	2.945
33	Jason Preston	2.89
34	Rokas Jokubaitis	2.835
35	Herbert Jones	2.78
36	Miles McBride	2.725
37	JT Thor	2.67
38	Ayo Dosunmu	2.615
39	Neemias Queta	2.56
40	Jared Butler	2.505
41	Joe Wieskamp	2.45
42	Isaiah Livers	2.395
43	Greg Brown	2.34
44	Kessler Edwards	2.285
45	Juhann Begarin	2.23
46	Dalano Banton	2.175
47	David Johnson	2.12
48	Sharife Cooper	2.065
49	Marcus Zegarowski	2.01
50	Filip Petrusev	1.955
51	Brandon Boston	1.9
52	Luka Garza	1.845
53	Charles Bassey	1.79
54	Sandro Mamukelashvili	1.735
55	Aaron Wiggins	1.68
56	Scottie Lewis	1.625
57	Balsa Koprivica	1.57
58	Jericho Sims	1.515
59	RaiQuan Gray	1.46
60	Georgios Kalaitzakis	1.405

5-Templates **(30%)**

Write a template for a function called countItemFrequency that accepts as parameters a vector and a value which may be contained in the vector. Iterate through the vector and count the number of occurrences of the value in the vector and return the count to the user. with a custom class with the == operator overloaded.

You have to input 15(int) and A(char), and output the number of occurrences of the value in the vector.



```
16
14
A
6
15
A
20
16
C
24
13
15
C
19
D
24
15
24
B
24
15
23
15
6
12
8
17
19
B
33
15
14
D
15
8
B
12
18
C
15
```

(10%)

input:15

output:8(the number of occurrences of the value)

(10%)

input:A

output:2(the number of occurrences of the value)