소프트웨어설계 문제 제작

2019312014 박병준

1. 클래스&객체 문제(난이도 5)

문제

클래스를 이용해 문제 블랙잭 게임을 구현해야 한다. 블랙잭은 카드의 합이 21 을 넘지 않는 한도 내에서, 카드의 합을 최대한 크게 만드는 게임이다. 블랙잭은 트럼프 카드를 사용한다. 카드에는 스페이드, 다이아, 하트, 클로버 4 종류의 무늬가 있으며, 각 무늬별로 13 종류(2~10, A, J, Q, K)의 카드가 있다. 즉, 카드는 총 52 장이다. 카드 중 A 카드는 이전에 받은 카드의 합이 10 이하일 경우 11 을 나타내고, 이전에 받은 카드의 합이 11 이상일 경우에는 1 을 나타낸다. J 카드는 11, Q 카드는 12, K 카드는 13 을 나타낸다.

블랙잭 관련 많은 규칙들이 있지만, 여기서는 플레이어와 딜러 간의 일대일 대결로만 게임이 이루어진다. 플레이어는 시드 머니를 가지고 있다. 게임 진행 시, 플레이어는 갖고 있는 시드 머니 한도에서 원하는 만큼 배팅한다. 그 후, 딜러는 2 장의 카드를 자신에게 할당하고, 플레이어에게 3 장의 카드를 할당한다. 카드 분배가 다 끝나면, 승패를 결정한다. 딜러와 플레이어 모두 자신의 카드의 합이 21 을 초과하면 무승부이다. (카드의 합이 21 을 초과한 경우를 burst 라 한다.)딜러와 플레이어 둘 중 한 사람만 자신의 카드의 합이 21 을 초과하면, 자신의 카드의 합이 21 을 초과하지 않은 사람이 승리한다. 자신의 카드의 합이 21 을 초과한 사람이 없다면, 딜러의 카드 2 장의 합과 플레이어가 받은 3 장의 카드의 합을 비교하여 21 에 더 근접한 사람이 승리한다. 플레이어가 승리하면 배팅한 만큼 돈을 획득할 수 있다. 무승부이면 플레이어는 돈을 잃지도 얻지도 않는다. 딜러가 승리하면 플레이어는 배팅한 만큼 돈을 잃게 된다. (Goorm oj 의 한계로

인해) 게임은 단판으로만 진행되며, 기존 규칙과 달리 플레이어는 카드 3 장을 받은 후 추가로 카드를 더 받을 수 없다.

입력

플레이어 정보(이름, 나이, 시드머니)를 순서대로 한줄씩 입력한다. 나이와 시드머니는 0 보다 큰 정수를 입력해야 한다. 그리고 플레이어가 배팅할 액수를 입력한다. 배팅할 액수로 정수를 입력해야 하며, 시드머니를 초과할 수 없다. 이후, 딜러가 받을 카드 숫자 2 개를 입력한다. (1 이상 13 이하의 정수를 입력해야 한다. 1 입력 시 이전에 받은 카드의 합이 10 이하라면 승패 결정을 위한 합계 계산 시 A 카드의 숫자는 11 로 인식된다. 그러나 J 카드와는 다른 종류의 카드임을 유의해야 한다. 1 입력 시 이전에 받은 카드의 합이 10 초과라면 카드의 숫자는 1 로 인식된다.) 플레이어가 받을 카드 숫자 3 개를 입력한다. (동일한 종류의 카드는 게임이 진행되는 동안 4 개까지만 배부될 수 있으며, 4 개를 초과해서 배부할 수 없다. 4 개를 초과할 시 다른 종류 카드 숫자 하나를 입력받을 때까지 재입력 요구를 받게 된다.)

출력

플레이어의 정보가 입력된 후, 정보 확인 차원에서 플레이어 정보를 이름, 나이, 시드머니 순으로 한줄씩 출력한다. 플레이어가 배팅할 액수가 입력된 뒤, 시드머니에서 배팅한 금액을 뺀 나머지 금액을 출력한다. 카드 분배에서 같은 종류의 카드 숫자를 게임에서 5 개 이상 입력 받을 시, “choose another value”를 출력하여 다른 숫자의 카드로 재입력을 유도한다. 카드 분배(딜러에게 2 장, 플레이어에게 3 장 배분)가 모두 끝난 뒤, 딜러의 카드 숫자(2 장) 합과 플레이어의 카드 숫자(3 장) 합을 한줄씩 출력한다. 게임 승무패에 따른 플레이어의 돈을 최종 출력한다.

테스트 케이스 1(player win)

입력 1

minsoo

32

500

50

5 5

7 9 1

출력 1

minsoo

32

500

450

10

17

550

테스트 케이스 2(burst 없는 draw)

입력 2

minsoo

32

500

50

9 1

5 7 8

출력 2

minsoo

32

500

450

20

20

500

테스트 케이스 3(draw 둘다 burst)

입력 3

minsoo

32

500

50

1 13

12 13 13

출력 3

minsoo

32

500

450

24

38

500

테스트 케이스 4(player lose)

입력 4

minsoo

32

500

50

10 9

7 2 5

출력 4

minsoo

32

500

450

19

14

450

테스트 케이스 5(player 만 burst)

입력 5

minsoo

32

500

50

2 8

1 13 7

출력 5

minsoo

32

500

450

10

31

450

테스트 케이스 6(dealer 만 burst)

입력 6

minsoo

32

500

50

13 9

6 7 8

출력 6

minsoo

450

22

21

550

테스트 케이스 7(1 11 섞어 입력)

입력 7

skku

600

50000

45678

12 1

13 11 11

출력 7

skku

600

50000

4322

13

35

4322

테스트 케이스 8

입력 8

minsoo

32

500

50

7 8

7 6 3

출력 8

minsoo

32

500

450

15

16

550

테스트 케이스 9(같은 종류 카드 5 개 이상 분배)

입력 9

minsoo

32

500

50

4 4

4 4 4

7

출력 9

minsoo

32

500

450

choose another value

8

15

550

테스트 케이스 10(A 5 개 이상 경우)

입력 10

minsoo

32

500

121

1 1

1 1 1

1

1

11

출력 10

minsoo

32

500

379

choose another value

choose another value

choose another value

12

23

379

테스트 케이스 11(J 5 개 이상 경우)

입력 11

minsoo

32

500

50

11 11

11 11 11

11

11

1

출력 11

minsoo

32

500

450

choose another value

choose another value

choose another value

22

23

500

예시 코드

```
#include <iostream>

#include <string>

using namespace std;



//예시 코드에선 클래스 멤버 함수의 구현을 클래스 안에서 했습니다. 예시 코드와 달리 멤버 함수의 선언과 구현을 따로 해도 무방합니다.

class DeckofCards // main 함수에서 DeckofCards deck[14]로 선언하고 사용. 덱에서 종류별 카드 개수가 몇개인지 확인

{

private:

    int count; // 종류별 카드 개수

public:

    DeckofCards()

    {

        //구현

    }

    int getCount()

    {
```

```
        //구현
    }
    void addCount(int num)
    {
        //구현
    }
    int cardDistribute(DeckofCards *deck, int value)
    {
        if (count >= 4)
        {
            //구현
        }
        else
        {
            //구현
        }
    }
};
```

```
class Person
{
private:
    string name;
    int age;
    int money;
    DeckofCards myCard[14];
public:
    Person()
    {
        //구현
    }
    Person(string name, int age, int money)
    {
        //구현
    }
    string getName()
    {
        //구현
```

```
}

int getAge()

{

    //구현

}

int getMoney()

{

    //구현

}

int subMoney(int money)

{

    //구현

}

int addMoney(int money)

{

    //구현

}

bool isBurst() // 카드 합이 21 초과되어 burst 된 경우 true return

{
```

```
        //구현
    }
    void addCard(int num) // card 를 받는다.
    {
        if (num == 1)
        {
            //구현
        }
        else if (num == 11)
        {
            //구현
        }
        myCard[num].addCount(1);
    }
    int myCardPoint() // card 점수를 보여준다.
    {
        //구현
    }
};
```

```
int main()
{
    string name;
    int age, money;
    //구현

    Person player(name, age, money);
    //구현

    //게임 시작

    // 배팅금액 입력

    // 카드 분배(딜러 2 개, 플레이어 3 개)
    Person dealer;
    DeckofCards deck[14]; // 전체 카드 덱. 무늬별 카드 개수 4 개 초과 시 분배 억제. 다른 숫자 카드로 분배 유도. deck[0]은 사용하지 않는다.
    //구현
```

```
    // 최종 카드 합 확인
    cout << dealer.myCardPoint() << endl;
    cout << player.myCardPoint() << endl;

    //burst 확인

    // 승패 결정

    // 배팅금액 계산

    return 0;
}
```

2. 객체 포인터&동적 생성 문제(난이도 3)

문제

문자열 저장하는 덱(Deque)을 구현한 다음, 입력으로 주어지는 명령을 처리하는 프로그램을 작성해야 한다. 예시 코드를 기반으로 Deque 클래스를 반드시 사용해야 한다. 명령은 총 여덟 가지이다.

- push_front X: 문자열 X 를 덱의 앞에 넣는다.

- push_back X: 문자열 X 를 덱의 뒤에 넣는다.

- pop_front: 덱의 가장 앞에 있는 문자열을 빼고, 그 문자열을 출력한다. 만약, 덱에 들어있는 문자열이 없는 경우에는 -1 을 출력한다.

- pop_back: 덱의 가장 뒤에 있는 문자열을 빼고, 그 문자열을 출력한다. 만약, 덱에 들어있는 문자열이 없는 경우에는 -1 을 출력한다.

- size: 덱에 들어있는 문자열의 개수를 출력한다.

- empty: 덱이 비어있으면 1 을, 아니면 0 을 출력한다.

- front: 덱의 가장 앞에 있는 문자열을 출력한다. 만약 덱에 들어있는 문자열이 없는 경우에는 -1 을 출력한다.

- back: 덱의 가장 뒤에 있는 문자열을 출력한다. 만약 덱에 들어있는 문자열이 없는 경우에는 -1 을 출력한다.

입력

첫째 줄에 명령의 수 n (1 ≤ n ≤ 10,000)이 주어진다. 덱은 크기 n 만큼 동적으로 생성된다. 둘째 줄부터 n 개의 줄에는 명령이 하나씩 주어진다. 문자열은 공백은 공백 없이 입력된다. q 를 입력하면 프로그램이 즉시 종료된다. 문제에 나와있지 않은 명령이 주어지는 경우는 없다.

출력

출력해야하는 명령이 주어질 때마다, 한 줄에 하나씩 출력한다.

테스트 케이스 1

입력 1

15

push_back as

push_front asd

front

back

size

empty

pop_front

pop_back

pop_front

size

empty

pop_back

push_front asdf

empty

front

출력 1

asd

as

2

0

asd

as

-1

0

1

-1

0

asdf

테스트 케이스 2

입력 2

22

front

back

pop_front

pop_back

push_front airplane

front

pop_back

push_back dog

back

pop_front

push_front universe

push_front cat

front

back

pop_back

pop_back

push_back cellphone

push_back 1234

front

back

pop_back

pop_back

출력 2

-1

-1

-1

-1

airplane

airplane

dog

dog

cat

universe

universe

cat

cellphone

1234

1234

cellphone

테스트 케이스 3

입력 3

10000

q

출력 3

(없음)

테스트 케이스 4

입력 4

4

push_front son

push_front cha

push_front park

pop_back

출력 4

son

테스트 케이스 5

입력 5

5

push_front apple

pop_back

push_back car

pop_front

size

출력 5

apple

car

0

테스트 케이스 6

입력 6

2

push_front ball

push_back car

출력 6

(없음)

테스트 케이스 7

입력 7

9

push_front sage

push_front exile

push_back jerk

push_back pretenses

back

push_front warden

back

pop_back

back

출력 7

pretenses

pretenses

pretenses

jerk

테스트 케이스 8

입력 8

4

push_front son

push_front kim

pop_front

front

출력 8

kim

son

테스트 케이스 9

입력 9

2

push_back cho

front

출력 9

cho

테스트 케이스 10

입력 10

10

push_back abide

push_back siege

push_front hefty

pop_front

front

pop_back

back

size

empty

pop_back

출력 10

hefty

abide

siege

abide

1

0

abide

테스트 케이스 11

입력 11

17

push_back 10

push_front 20

push_back 30

size

pop_front

pop_back

size

push_front 40

empty

pop_back

empty

front

back

size

q

출력 11

3

20

30

1

0

10

0

40

40

1

예시 코드

```
#include <iostream>
#include <string>
using namespace std;

class Deque
{
private:
	int *deque;
	int begin, end;

public:
	Deque(int n)
```

```
    {
        //작성 필요
    }

    void push_front(string s);
    void push_back(string s);
    void pop_front();
    void pop_back();
    void size();
    bool empty();
    void front();
    void back();

    ~Deque()
    {
        //작성 필요
    }
};
```

```
//함수 구현 필요


int main()
{
	int n;
	cin >> n;
	Deque d(n);


	for (int i = 0; i < n; i++)
	{
		string cmd;
		cin >> cmd;


		if (cmd=="q")
		{
			break;
		}


		if (cmd == "push_front")
```

```
{
	string s;
	cin >> s;

	d.push_front(s);
}

else if (cmd == "push_back")
{
	string s;
	cin >> s;

	d.push_back(s);
}

else if (cmd == "pop_front")
{
	d.pop_front();
}
```

```
else if (cmd == "pop_back")
{
	d.pop_back();
}

else if (cmd == "size")
{
	d.size();
}

else if (cmd == "empty")
{
	cout << d.empty() << endl;
}

else if (cmd == "front")
{
	d.front();
```

```
        }



        else if (cmd == "back")

        {

            d.back();

        }

    }



    return 0;

}
```