〈자료구조 및 실습〉 퀴즈 문제지 (A형)

2020.04.29 (수)

학번	학과	이름

※ 문제에 대한 안내

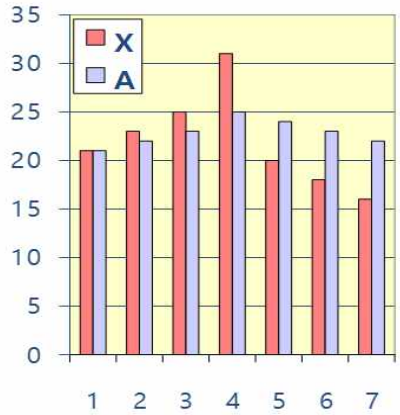
- 문제지는 총 6페이지이고, **문제의 순서는 난이도와 관계없다.**
- 특별한 언급이 없으면 문제의 조건에 맞지 않는 입력은 입력되지 않는다고 가정하라.
- 특별한 언급이 없으면, 각 줄의 맨 앞과 맨 뒤에는 공백을 출력하지 않는다.
- 출력 예시에서 □는 각 줄의 맨 앞과 맨 뒤에 출력되는 공백을 의미한다.
- 입출력 예시에서 ↦ 이 후는 각 입력과 출력에 대한 설명이다.

[문제 1](50점) 누적 평균

원시 데이터값들로 구성된 배열 **X**의 **i**-번째 **누적평균**(prefix average)이란 **X**의 **i**-번째에 이르기까지의 (**i** + **1**)개 원소들의 평균이다. 즉,

A[**i**] = (**X**[**0**] + **X**[**1**] + … + **X**[**i**])/(**i** + **1**)

누적평균은 경제, 통계 분야에서 오르내림 변동을 순화시킴으로써 대략적 추세를 얻어내기 위해 사용된다. 일례로 부동산, 주식, 펀드 등에도 자주 활용된다.. 이 문제는 배열 **X**의 누적평균(prefix average) 배열 **A**를 구하는 프로그램을 구현하고 테스트하는데 관한 것이다.



- 아래 의사코드 함수 **prefixAverages1**은 위 정의를 있는 그대로 이용하여 누적평균값들을 **2**차 시간에 구한다.

※ 참고: 각 명령문 오른 편 중괄호 내의 수식은 실행시간 분석을 위한 근거로서, 해당 명령문이 수행하는 치환, 반환, 산술 및 비교 연산 등 기본 명령들의 수행 횟수를 나타낸다.

```
Alg prefixAverages1(X, n)      {ver.1}
   input array X of n integers
   output array A of prefix averages of X

1. for i ← 0 to n - 1          {n}
      sum ← 0                  {n}
      for j ← 0 to i           {1 + 2 + ··· + n}
         sum ← sum + X[j]      {1 + 2 + ··· + n}
      A[i] ← sum/(i + 1)       {n}
2. return A                    {1}
                               {Total O(n²)}
```

- 위의 의사코드 함수 **prefixAverages1**의 내용을 살펴보면, **i** 번째 외부 반복에서는, 바로 전 **i** – **1**번째 반복에서 구했던 [**0** ~ **i** – **1**]의 합에, **i** + **1** 번째 원소 값 한 개만을 더해 현재 합을 얻어 평균을 구한다. 따라서 이를 수정하여 이전의 **i** – **1**번째까지의 합을 보관하여 다음 반복으로 전달하는 방식으로 반복한다면 현재 합을 구하는데 필요한 시간을 단축할 수 있다는 것을 알 수 있다. 이렇게 중간 합을 보관하는 방식으로 알고리즘을 개선한 함수 **prefixAverage2**는 누적평균값들을 선형 시간에 구할 수 있게 된다.

- 함수 **prefixAverages1**과 **prefixAverages2**, 그리고 이들을 테스트할 수 있는 **main** 함수를 구현하여 아래 테스트를 수행하라.

입출력 형식:

1) **main** 함수는 아래 형식의 표준입력을 사용하여 배열 **X**를 초기화한 후 각 함수를 호출한다.

입력 : **main** 함수는 다음 값들을 표준입력 받는다.

첫 번째 라인: 정수 **n** (배열 **X**의 크기)

두 번째 이후 라인: **X[0] X[1] X[2]** … (배열 **X**, 한 라인 상의 양의 정수 수열)

※ **힌트: n**의 크기에는 제한이 없다. 따라서 동적 할당을 사용하여야 함)

2) **main** 함수는 아래 형식의 표준출력을 사용하여 각 함수로부터 반환된 배열 **A**를 출력한다.

출력 : **A[0] A[1] A[2]** …

(배열 **X**와 같은 크기의 배열 **A**의 원소들을 나타내는 한 라인 상의 양의 정수 수열로서 첫 번째 라인은 **prefixAverages1**의 출력을, 두 번째 라인은 **prefixAverages2**의 출력을 나타낸다)

3) 평균 계산 시 소수점 이하를 반올림하여 정수로 구한다. 정확한 반올림을 위해, %.f를 쓰지 말고 int 성질을 이용해서 반올림하라.

입력 예시 1

```
3          ↦ 배열 X 크기
5 1 9      ↦ 배열 X
```

출력 예시 1

```
5 3 5      ↦ prefixAverages1의 출력
5 3 5      ↦ prefixAverages2의 출력
```

입력 예시 2

```
6                   ↦ 배열 X 크기
1 3 2 10 6 8        ↦ 배열 X
```

출력 예시 2

```
1 2 2 4 4 5    ↦ prefixAverages1의 출력
1 2 2 4 4 5    ↦ prefixAverages2의 출력
```

[문제 2](50점) 이진변환기

재귀를 이용하여 10진수를 2진수로 변환하여 출력하시오. **(반복문 사용시 감점 0점처리)**

- **입력은 10진수 정수 n이 입력된다.**

입력 예시 1

```
7
```

출력 예시 1

```
1 1 1
```

[문제 3](100점) N x M (1≤N, M≤100) 크기의 행렬에 1 ~ MN 의 수를 아래 그림과 같이 나선형으로 채운 결과를 출력하시오.

4 x 4 행렬 :

1	2	3	4
12	13	14	5
11	16	15	6
10	9	8	7

4 x 5 행렬 :

1	2	3	4	5
14	15	16	17	6
13	20	19	18	7
12	11	10	9	8

입력 예시 1

```
4 5          ↦ 행렬 크기 N, M
```

출력 예시 1

```
□1 2 3 4 5        ↦ 한 줄에 한 행씩 출력
□14 15 16 17 6
□13 20 19 18 7
□12 11 10 9 8
```

[문제 4](100점) N x M (1≤N, M≤100) 크기의 행렬에 1 ~ MN 의 수를 아래 그림과 같이 ↙ 대각선 방향으로 채운 결과를 출력하시오.

4 x 4 행렬 :

1	2	4	7
3	5	8	11
6	9	12	14
10	13	15	16

4 x 5 행렬 :

1	2	4	7	11
3	5	8	12	15
6	9	13	16	18
10	14	17	19	20

입력 예시 1

```
4 5          ↦ 행렬 크기 N, M
```

출력 예시 1

```
□1 2 4 7 11       ↦ 한 줄에 한 행씩 출력
□3 5 8 12 15
□6 9 13 16 18
□10 14 17 19 20
```

[문제 5](150점) 위의 설명과 같이 다항식을 헤더 단일연결리스트로 표현하고, 다항식의 덧셈을 구하는 프로그램을 작성하라.

- 입력에 대한 설명 (아래 입출력 예시 참조)
  - 첫 번째 다항식의 항의 개수가 입력되고, 이후에 다항식의 각 항의 (계수, 지수) 쌍이 지수의 내림차순으로 입력됨
  - 동일한 방식으로 두 번째 다항식의 정보가 입력됨

- 출력에 대한 설명 (아래 입출력 예시 참조)
  - 결과 다항식의 각 항의 (계수, 지수) 쌍을 지수의 내림차순으로 출력

입력 예시 1

```
3          ↦ 첫 번째 다항식의 항의 개수
5 3 3 2 3 1   ↦ 5x^3 + 3x^2 + 3x
3          ↦ 두 번째 다항식의 항의 개수
2 6 2 3 1 0   ↦ 2x^6 + 2x^3 + 1
```

출력 예시 1

```
□2 6 7 3 3 2 3 1 1 0  ↦ 2x^6+7x^3+3x^2+3x+1
```

입력 예시 2

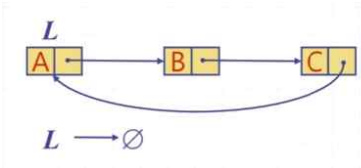
```
2          ↦ 첫 번째 다항식의 항의 개수
2 7 3 0     ↦ 2x^7 + 3
3          ↦ 두 번째 다항식의 항의 개수
-3 10 3 7 -3 0   ↦ -3x^10 + 3x^7 -3
```

출력 예시 2

```
□-3 10 5 7    ↦ -3x^10 + 5x^7
```

[문제 6](250점) 원형 리스트 구현

1. 원형 연결리스트 구조



○ 각 노드에 저장되는 정보

- elem: 원소
- next: 다음 노드를 가리키는 링크

2. 원형 연결리스트에 사용되는 함수 정의

```
Alg size()
1. return (N − f + l + 1) % N
```

```
Alg get(r)
1. if ((r < 0) || (r ≥ size()))
       invalidRankException()
2. return A[(f + r) % N]

Alg set(r, e)
1. if ((r < 0) || (r ≥ size()))
       invalidRankException()
2. A[(f + r) % N] ← e
3. return e
```

```
Alg add(r, e)
1. n ← size()
2. if (n = N − 1)          {reserve 1 cell}
       fullListException()
3. if ((r < 0) || (r > n))
       invalidRankException()
4. if (r < n/2)
       for i ← f to (f + r − 1) % N
           A[(i − 1) % N] ← A[i]
       A[(f + r − 1) % N] ← e
       f ← (f − 1) % N
   else
       for i ← l downto (f + r) % N
           A[(i + 1) % N] ← A[i]
       A[(f + r) % N] ← e
       l ← (l + 1) % N
5. return
```

```
Alg remove(r)
1. n ← size()
2. if (n = 0)
       emptyListException()
3. if ((r < 0) || (r ≥ n))
       invalidRankException()
4. e ← A[(f + r) % N]
5. if (r < n/2)
       for i ← (f + r − 1) % N downto f
           A[(i + 1) % N] ← A[i]
       f ← (f + 1) % N
   else
       for i ← (f + r + 1) % N to l
           A[(i − 1) % N] ← A[i]
       l ← (l − 1) % N
6. return e
```

위에서 설명한 이중연결리스트를 구현하고, 영문자 리스트 ADT를 구현하시오.

○ 다음 네 가지 연산을 지원해야 함 (위치는 1번째부터 시작한다고 가정)

- add(list, position, item) : list의 position번째에 item을 추가한다.
- delete(list, position) : list의 position번째 위치한 item을 삭제한다.
- get_entry(list, position) : list의 position번째 위치한 item 값을 리턴한다.
- print(list) : list의 모든 item을 list에 저장된 순서대로 공백 없이 출력한다.

※ position 정보가 유효하지 않으면 화면에 에러 메시지 "invalid position" 출력하고, 해당연산은무시한다.

○ 입력에 대한 설명 (아래 입출력 예시 참조)

- 각 연산의 내용이 한 줄에 하나씩 입력되고, 하나의 줄에는 연산의 종류, 위치, 아이템이 차례로 입력된다.
- 연산의 종류: 연산 이름의 맨 앞 영문자가 대문자 (A, D, P)로 주어진다.
- 위치: 양의 정수
- 아이템: 영문자 (대문자, 소문자 모두 가능)

입력 예시 1		출력 예시 1
18	↦ 연산의 개수: 18	0 1 2 3 4 5
A 1 0	↦ add(list, 1, '0')	1 2 5
A 2 1	↦ add(list, 2, '1')	7 8 9
A 3 2	↦ add(list, 3, '2')	
A 4 3	↦ add(list, 4, '3')	
A 5 4	↦ add(list, 5, '4')	
A 6 5	↦ add(list, 6, '5')	
P	↦ print(list)	
D 1	↦ delete(list,1)	
D 4	↦ delete(list,4)	
D 3	↦ delete(list,3)	
P	↦ print(list)	
A 1 9	↦ add(list, 1, '9')	
A 1 8	↦ add(list, 1, '8')	
A 1 7	↦ add(list, 1, '7')	
D 6	↦ delete(list,6)	
D 5	↦ delete(list,5)	
D 4	↦ delete(list,4)	
P	↦ print(list)	