컴퓨터공학설계및실험I
기말 프로젝트
(2020년 12월 8일)

※ 주의사항

주의사항을 읽지 않았을 때의 불이익은 학생 책임이므로, 반드시 읽어보시기 바랍니다.

1. 모든 소스 코드에 적절한 주석을 반드시 달 것.
2. 사용 언어는 **C언어**만 사용할 것. (*** **C++ 사용 금지** ***)
3. 문제별로 주어진 함수의 프로토타입을 바꾸지 말 것.
4. 모든 자료구조는 직접 구현할 것. **(*** STL 사용 금지 ***)**
5. 한 문제당 **한 개의 C파일**로 저장한 후, 파일 이름을 학번_P(문제번호).c 로 생성할 것. (ex. 20201234_P1.c)
6. 채점 환경은 CSPRO이므로, 제출 전 반드시 점검할 것. (Makefile은 제출하지 않아도 됨)

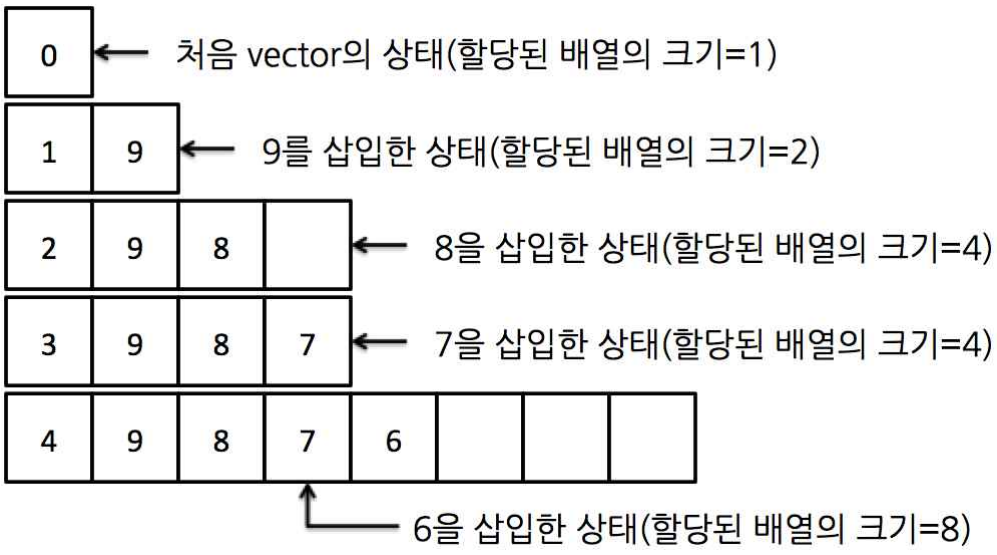
1. Vector [Stack, 50점]

C++에는 Vector라는 자료구조가 있다. Vector는 가변 길이 배열로써 배열의 크기가 배열에 들어있는 원소의 개수에 따라서 변한다. 예를 들어, 현재 동적 배열의 크기가 4이고 원소의 개수가 4개인 상황에서, 배열에 새로운 원소를 삽입하게 되면 현재 할당된 배열의 크기를 두 배 만큼 증가시킨다. 그 결과, 동적 배열의 크기는 8로 증가하게 되고, 원소의 개수는 5개가 되는 것이다. 삭제의 경우에도 삽입과 비슷하다. 현재 동적 배열의 크기가 8이고 원소의 개수가 5개일 때, 배열의 원소를 하나 삭제하면 동적 배열의 크기를 절반으로 감소시킨다. 그 결과, 동적 배열의 크기는 4로 줄어들게 되고, 원소의 개수는 4개가 되는 것이다.

Vector 자료구조는 다음과 같은 특성을 갖는다.

① Vector에 원소를 삽입할 때에는 **맨 뒤**에 추가한다.
② Vector의 원소를 삭제할 때에는 **맨 뒤의 원소**를 삭제한다.
③ Vector의 0번째 index는 Vector에 있는 **원소의 개수**를 나타낸다.
④ Vector의 원소는 1번 index부터 시작한다.
⑤ Vector를 생성할 때에는 1차원 정수형 포인터를 크기 1로 할당하고, Vector가 비어있으므로 0번 index의 값을 0으로 초기화한다.

이해를 돕기 위해, 처음 Vector를 생성하고 9, 8, 7, 6을 차례로 삽입하는 것을 그림으로 표현하면 아래와 같다.



당신은 위에서 설명한 Vector를 다음의 함수를 작성하여 구현하고, 테스트하는 프로그램을 작성해야 한다. 테스트 프로그램은 아래의 명령어를 숫자로 입력받아 해당 연산을 수행할 수 있도록 만든다.

명령어	호출하는 함수	기능
0	empty	0을 입력받으면 Vector가 비어있는지 확인하여 비어있으면 1, 그렇지 않으면 0을 출력한다.
1	push	1을 입력받으면 Vector에 추가할 값을 하나 더 입력받고 Vector 맨 뒤에 원소를 하나 추가한 후 할당된 배열의 크기를 출력한다.
2	pop	Vector의 가장 뒤 원소를 삭제하고, 이를 출력한 후 할당된 배열의 크기를 출력한다. 만약 Vector가 비어있으면 -1을 출력한다.
3	front	Vector의 가장 앞 원소를 출력한다. 만약 Vector가 비어있으면 -1을 출력한다.
4	back	Vector의 가장 뒤 원소를 출력한다. 만약 Vector가 비어있으면 -1을 출력한다.
5	print_all	Vector의 모든 원소를 출력한다. 만약 Vector가 비어있으면 -1을 출력한다.

※ 사용해야 하는 함수 원형(Prototype)

```
int empty(int *a);
void push(int **a, int *element_size, int *alloc_size, int value);
int pop(int **a, int *element_size, int *alloc_size);
int front(int *a, int element_size);
int back(int *a, int element_size);
int print_all(int *a, int element_size);
```

이 6가지 함수는 **반드시** 구현해야 하고, 필요할 시 함수를 추가하는 것은 가능하다.

※ 함수 및 파라미터 설명

① empty 함수는 Vector a가 비어있으면 1을, 그렇지 않으면 0을 반환한다.
② push 함수는 Vector a의 맨 뒤에 value를 삽입한다. 이 때, Vector a의 할당된 크기(alloc_size)가 가득 찬다면 위에서 설명한 대로 원소의 개수에 맞추어 realloc 해준다.
③ pop 함수는 Vector a의 맨 뒤 원소를 삭제한다. 이 때, Vector a의 할당된 크기(alloc_size)와 원소의 개수(element_size)를 위에서 설명한 대로 크기를 줄여준다. 만약 Vector가 비어있으면 -1을 반환한다.
④ front 함수는 Vector a의 가장 앞 원소(1번째 index)를 반환한다. 만약 Vector a가 비어있으면 -1을 반환한다.
⑤ back 함수는 Vector a의 가장 뒤 원소를 반환한다. 만약 Vector a가 비어있으면 -1을 반환한다.
⑥ print_all 함수는 Vector a의 원소를 전부 standard output으로 출력한다. 이 때, Vector의 0번 index는 원소가 아니라 Vector a에 들어있는 원소의 개수를 의미하므로 **제외하고** 출력한다. 만약 Vector가 비었으면 -1을 반환한다.

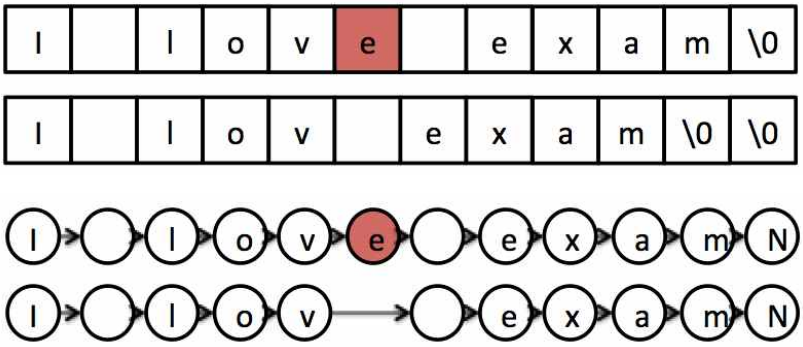
※ 제한 사항
① 정적 배열 및 전역 변수 사용을 금지한다.
② <stdio.h>, <stdlib.h>를 제외한 헤더는 사용을 금지한다.

※ 입출력 예시

(입력1)	0
(출력1)	1
(입력2)	1 1
(출력2)	2
(입력3)	1 2
(출력3)	4
(입력4)	1 3
(출력4)	4
(입력5)	1 4
(출력5)	8
(입력6)	3
(출력6)	1
(입력7)	4
(출력7)	4
(입력8)	5
(출력8)	1 2 3 4
(입력9)	2
(출력9)	4 4
(입력10)	2
(출력10)	3 4
(입력11)	5
(출력11)	1 2

2. 1-Line Text Editor [Linked List, 50점]

Text Editor를 구현할 때는 보통 Linked List를 이용한다. 왜냐하면 문자 하나를 삭제했을 때, 그 문자 뒤에 있는 모든 문자열을 한 칸 앞으로 땡겨야 하는데, Linked List를 이용하면 노드 하나만 삭제하면 되기 때문이다. 예를 들어, 문자열 "I love exam"에서 처음 나오는 'e'를 삭제할 때 아래 그림은 배열의 비효율성을 보여주고, 그 다음 그림은 Linked List의 효율성을 보여준다.



당신은 한 줄로 된 간단한 Text Editor를 구현해야 한다. 이 Text Editor는 영어 소문자만을 기록할 수 있는 편집기로, 최대 1,000글자까지 입력할 수 있다. 이 편집기에는 커서라는 것이 있는데, 커서는 문장의 맨 앞(첫 번째 문자의 왼쪽), 문장의 맨 뒤(마지막 문자의 오른쪽), 또는 문장 중간의 임의의 곳(연속된 두 문자 사이)에 위치할 수 있다. 즉, 길이가 L인 문자열이 현재 편집기에 입력되어 있으면, 커서가 위치할 수 있는 곳은 L+1가지 경우가 있다. 이 편집기가 지원하는 명령어는 다음과 같다.

명령어	호출하는 함수	기능
L	left	커서를 왼쪽으로 한 칸 옮김(커서가 문장의 맨 앞이면 무시)
R	right	커서를 오른쪽으로 한 칸 옮김(커서가 문장의 맨 뒤라면 무시)
D	delete	커서 왼쪽에 있는 문자를 삭제함(커서가 문장의 맨 앞이면 무시)
A $	add	$라는 문자열을 커서 오른쪽에 추가함. 추가 후, 커서는 $의 오른쪽에 위치함
Q	save	프로그램 종료 후 문자열을 data.txt에 저장

커서를 왼쪽으로 옮길 수도 있기 때문에, Linked List에 다음 노드를 가리키는 포인터 뿐만이 아니라 이전 노드를 가리키는 포인터도 알아야 한다. 이를 Doubly Linked List라고 한다.

사용해야 할 구조체는 다음과 같다.

```
typedef struct node {
        char value;
        struct node* next;
        struct node* prev;
} NODE;
```

프로그램을 실행하고 나면 처음 입력으로 초기 문자열이 주어지고, 그 이후 입력한 명령어가 차례로 주어졌을 때, 해당 명령어를 입력했을 때의 결과를 즉각적으로 출력하는 프로그램을 작성하시오. 단, 초기 입력을 받고 난 후 명령어가 수행되기 전에 커서는 문장의 맨 뒤에 위치하고 있다.

※ 사용해야 하는 함수 원형(Prototype)

```
void left(NODE** cursor);
void right(NODE** cursor);
void del(NODE** cursor);
void add(NODE** cursor, char *data);
void quit(NODE* head);
```

이 5가지 함수는 반드시 구현해야 하고, 필요할 시 함수를 추가하는 것은 가능하다.

※ 함수 설명

① left 함수는 cursor를 받아 기존에 가리키던 노드에서 왼쪽의 노드를 가리키게끔 수정한다. (명령어 L에 대응)

② right 함수는 cursor를 받아 기존에 가리키던 노드에서 오른쪽의 노드를 가리키게끔 수정한다. (명령어 R에 대응)

③ del 함수는 cursor를 받아 cursor가 가리키는 노드의 왼쪽에 있는 노드를 삭제한다. 이 때, 삭제로 인해 커서가 한 칸 왼쪽으로 이동한 것처럼 생각할 수 있지만, 실제로 커서의 오른쪽에 있는 문자는 그대로이다. (명령어 D에 대응)

④ add 함수는 cursor를 받아 cursor가 가리키는 노드의 오른쪽에 새로운 노드를 삽입한다. 이 때, 추가하는 문자가 한 개가 아닐 수 있으므로 추가하는 문자의 개수만큼 노드를 새로 생성해야 한다. 최대 100글자까지 삽입할 수 있다. (명령어 A에 대응)

⑤ quit는 Linked List의 head 노드를 받아 마지막 노드까지 순차적으로 탐색하며 데이터를 텍스트 파일에 저장한다. 이 때, 텍스트 파일의 이름은 **data.txt** 로 한다. (명령어 Q에 대응)

※ 제한 사항

① 전역 변수 사용을 금지한다.

② <stdio.h>, <stdlib.h>를 제외한 헤더는 사용을 금지한다.

※ 입출력 예시 1

(입력1)	abcd
(입력2)	A x
(입력3)	L
(입력4)	A y
(입력5)	Q
(프로그램 종료)	
data.txt	abcdyx

※ 입출력 예시 2

(입력1)	isomorphiii
(입력2)	D
(입력3)	D
(입력4)	A sm
(입력5)	Q
(프로그램 종료)	
data.txt	isomorphism

※ 입출력 예시 3

(입력1)	cse lab
(입력2)	L
(입력3)	L
(입력4)	L
(입력5)	L
(입력6)	R
(입력7)	D
(입력8)	Q
(프로그램 종료)	
data.txt	cselab