텍스트 에디터

제한시간 30000 ms

키보드 입력은 char형태의 ASCII code로 주어지게 됩니다.
ASCII code는 printing character와 non-printing character로 나뉩니다.
printing character : 문자(영문), 숫자, 기호, space
non-printing character : Enter(LF/CR), BackSpace, Shift in/out, ESC, Ctrl+c/x/v, Arrow key 등

ASCII(6) : Ctrl+f
ASCII(5) : ENQ
ASCII(27) : ESC
ASCII(3,22,24) : Ctrl+c,v,x
ASCII(14~15) : Shift(out/in)
ASCII(17~20) : 방향키
ASCII(8) : BackSpace
ASCII(10) : LF
ASCII(32~126) : printing character
를 입력 받아 처리하도록 합시다.
Enter에 해당되는 입력은 CR없이 LF만 주어질 것입니다.
편의를 위해 LF(Line feed)가 CR(Carriage Return)의 동작까지 포함한다고 가정하겠습니다.

Text Editor의 제한 길이는 char 기준 100000입니다.

<cursor 이동 규칙>
right : line의 마지막일 경우 next line 처음으로 이동
left : line의 처음일 경우 previous line 마지막으로 이동
up : cursor가 가리키는 column index를 유지하되, previous line의 마지막이 column index보다 작다면 previous line의 마지막으로 이동, 첫 번째 line에서는 아무런 동작하지 않음
down : cursor가 가리키는 column index를 유지하되, next line의 마지막이 column index보다 작다면 next line의 마지막으로 이동, 마지막 line에서는 아무런 동작하지 않음

<영역지정>
shift in
방향키(여러번)
shift out
위순서의 입력에 따라 영역이 지정됩니다.
즉, Shift in 일 때의 cursor 위치에서 Shift out 일 때의 cursor 위치까지를 영역으로 지정하면 됩니다.
Shift in 과 out 사이에는 arrow key 이외의 키는 들어오지 않는다고 가정해도 좋습니다.
방향키에 의한 cursor 이동 규칙은 전 step과 완벽히 동일합니다.

영역지정 완료 직후 들어오는 입력에 따른 동작은 다음과 같습니다.
printing character : 해당영역을 입력된 ASCII code로 치환
방향키 : 영역지정 상태 해제 및 일반적인 방향키 동작
backspace : 해당영역을 삭제한 후 삭제된 영역의 앞 뒤를 이어줌
Enter(LF) : 해당영역을 삭제한 후 삭제된 영역의 앞 뒤를 이어주고 LF/CR을 수행. 위 case의 ACSII code가 들어오면 해당 동작 이후 영역지정 상태를 해제합니다. 그 이외의 키가 들어오면 영역지정 상태를 유지합니다.
Ctrl + x, c, v : 다음 step에서 설명합니다.

<복사 붙여넣기>
Ctrl + x : 잘라내기
지정된 영역을 잘라내기 한 후, 영역지정을 해제합니다.
영역지정이 되어있을 때만 동작합니다.

Ctrl + c : 복사하기
지정된 영역을 복사하기 한 후, 영역지정을 유지합니다.
영역지정이 되어있을 때만 동작합니다.

Ctrl + v : 붙여넣기
영역지정이 되어 있으면 :
지정된 영역을 삭제한 후, 잘라내거나 복사한 영역을 붙여넣기 합니다. 영역지정을 해제합니다.
영역지정이 되어 있지 않으면 :
현 cursor 위치에 잘라내거나 복사한 영역을 붙여넣기 합니다.
붙여넣기한 후의 cursor위치는 붙여넣기한 마지막 문자 다음을 가리키게 됩니다.

<찾기>
1. Ctrl + f
: 찾기 모드 시작

2. printing character (여러 번)
: '찾을 문자열'이며 space, enter(LF) 등을 포함할 수 있습니다.
찾기 모드일 때 입력 받는 printing character는 text editor에 write되지 않습니다.
찾을 문자열의 길이는 char기준 100을 넘지 않습니다.

3-1. ENQ (여러 번)
: 현재 cursor에서 시작하여 '찾을 문자열'을 찾아 cursor를 해당 문자열의 마지막으로 이동하고, 해당 문자열을 영역 지정한 상태가 됩니다. (step5 에서 shift로 영역 지정한 상태와 동일한 상태)
ENQ는 여러 번 연속으로 입력 받을 수 있으며 동작은 동일합니다.
text editor의 마지막 문자열을 찾은 이후에는 자동으로 문서의 처음부터 찾습니다.
중복된 문자열 찾기는 지원하지 않습니다. ex) 'ccc' 문자열에서 'cc'를 찾으면 한번만 찾아집니다.
text editor 전체를 찾아도 동일한 문자열이 없으면 찾기 모드를 해제합니다.

3-2. ESC
: 찾기 모드를 해제합니다.
만약 찾은 문자열을 영역 지정한 상태라면 영역 지정 상태를 유지합니다.

Ctrl+f 입력 후 ENQ나 ESC를 입력 받는 사이에 들어오는 인풋은 printing character, ENQ, Ctrl+X,C,V 뿐입니다.

<동작방식>
각 테스트 케이스의 처음에 init() 함수가 호출됩니다.
한 개의 input이 입력될 때마다 process() 함수가 한 번씩 호출됩니다.
하나의 테스트 케이스에서 process() 함수는 NULL이 입력될 때까지 계속 호출됩니다. (호출 횟수 제한은 없습니다.)
각 테스트 케이스의 마지막에 save()함수가 호출됩니다.
지금까지의 입력을 모두 처리한 결과(char*)를 save() 함수의 return 값으로 돌려주면 됩니다.

<자료구조>
향후 step에 대비하여 초기 자료구조를 배열이 아닌 "링크드 리스트"로 구현하는 것을 추천합니다.
ex)

```
class item {
  …
  char c;
  item* next;
  item* prev;
  …
}
```

<구현해야 할 함수 1>

```
void init();
// 각 테스트 케이스의 처음에 호출됩니다.
```

<구현해야 할 함수 2>

```
void process(char ASCII);
// 한 개의 input이 입력될 때마다 process() 함수가 한 번씩 호출됩니다.
```

<인자>
ASCII(6) : Ctrl+f
ASCII(5) : ENQ
ASCII(27) : ESC
ASCII(3,22,24) : Ctrl+c,v,x
ASCII(8) : BackSpace
ASCII(10) : LF
ASCII(14~15) : Shift(out/in)
ASCII(17~20) : 방향키
ASCII(32~126) : printing character

<구현해야 할 함수 3>

```
char* save();
// 각 테스트 케이스의 마지막에 호출됩니다.
```

<리턴 값>
지금까지의 입력을 모두 처리한 char array를 가리키는 pointer.
해당 char array 할당은 user code에서 하도록 합니다.

<input example>
1 // TC
65 // 'A'
66 // 'B'
65 // 'A'
6 // ctrl f
65 // 'A' : 찾을 문자열
5 // ENQ : 현재 cursor가 문서의 마지막이므로 첫 번째 'A'를 찾고 영역지정
5 // ENQ : 두번째 'A'를 찾고 영역지정
67 // 'C' : 지정된 영역이 'C'로 변경됨
0 // NULL : 현재 TC종료 신호

<위 예제에 대한 정답>
save()에서 return한 char pointer가 가리키는 곳의 array는 아래처럼 구성되어 있어야 합니다.
[0] ' '
[1] '!'
[2] 'A'
[3] 'a'
[4] '\0'

좀 더 자세한 사항은 main.cpp 를 읽고 분석해야 합니다.

main.cpp

```
#include <stdio.h>
#include "common.h"

static void printEdit(const char *a) {
	while (*a != NULL) {
		printf("%c", *a++);
	}
	printf("\n");
}

int main(void) {
	setbuf(stdout, NULL);

	int score = 0;
	int T;
	scanf("%d", &T);
	for (int t = 1; t <= T; t++) {
		init();

		int input = SPACE;
		while (true) {
			scanf("%d", &input);
			if (input == NULL) break;
			process((char)input);
		}
		char* userEditor = save();
		printf("\n#%d result : \n", t);
		printEdit(userEditor);
	}
	return 0;
}
```

common.h

```
#define LENMAX          (100000+1) // last +1 for '\0'
#ifndef NULL
#define NULL    0
#endif
#define SPACE  32
#define DEL     127
#define LF      10
#define BS      8
#define SHIFTOUT        14
#define SHIFTIN         15
#define ARROW_UP 17
#define ARROW_DOWN 18
#define ARROW_RIGHT 19
#define ARROW_LEFT 20
#define CTRL_C3
#define CTRL_F6
#define CTRL_V22
#define CTRL_X24
#define CTRL_Z26
#define ENQ     5
#define ESC     27

void init();
void process(char ASCII);
char* save();
```

user.cpp

```
void init(){

}

void process(char ASCII){

}

char *save(){

}
```