せっかくスタックを作ったのでキューも…

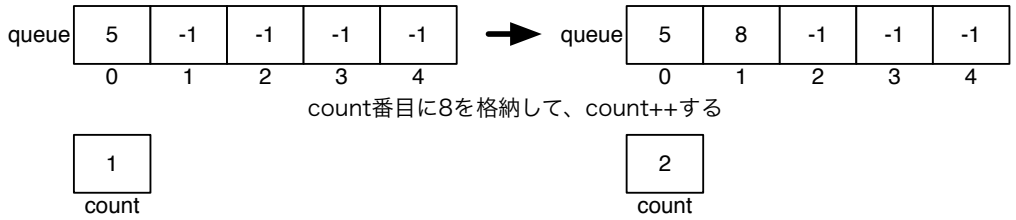
「キュー（Queue）」とは、スタックと同様にデータを管理する際によく使う構造です。スタックとキューの違いは、「データを格納する操作」のみで、スタックが配列の先頭に格納するのに対して、キューは**配列の最後の要素となるように格納します**。この格納する操作の違いによって、キューは「**先に入れた値**を先に取り出すことができる」という特徴を持ちます。（スタックは「**後に入れた値**を先にとりだすことができる」）

前回作ったスタックのプログラムを、次のように改良していくと完成します。

1. 配列 stack の名前を queue に変更する。（動作には影響ありませんが、気持ちが悪いので変更します。）
2. 関数 show() と clear() の中の配列も queue に変更して、そのまま使う。
3. キューに何個のデータが格納されているのかをカウントする変数 count をグローバル宣言する。
   （課題 1-4 で既に追加している場合は不要）
4. キューに格納する操作「エンキュー（enqueue）」を作る。

```
void enqueue(int x);
 /* 配列 queue の count 番目に引数 x を格納する */
 /* count++ する */
 /* （count が 5 以上の場合は、上記の操作を回避できるようにするとより良い） */
```

エンキューのイメージは次のようになる。

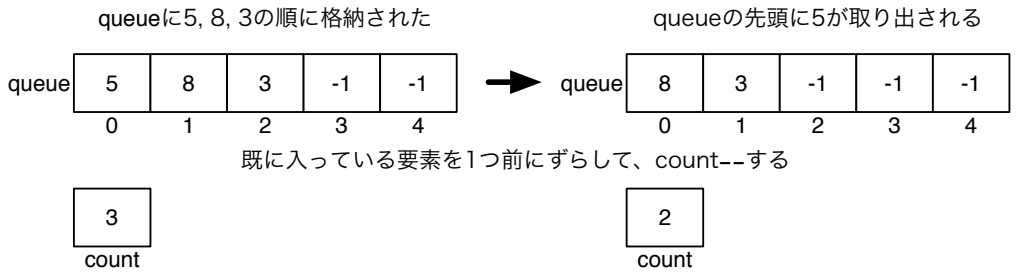



```
[main での処理]
 clear();
 enqueue(5);
 enqueue(8);
 enqueue(3);
 show();
[実行結果（5, 8, 3 の順にキューへ格納した）]
  5  8  3 -1 -1
```

5. キューから取り出す操作「デキュー（dequeue）」を作る。

```
int dequeue();
 /* スタックの pop() と全く同じ */
 /* return をする直前に、count--する */
 /* （count が 0 の場合は、上記の操作を回避できるようにするとより良い） */
```

デキューのイメージは次のようになる。




```
[main での処理（enqueue の続きに書いた場合）]
 printf("dequeue: %d\n", dequeue());
 show();
 printf("dequeue: %d\n", dequeue());
 show();
[実行結果（2 回 dequeue をして、5, 8 の順で取り出された)]
 dequeue: 5
  8  3 -1 -1 -1
 dequeue: 8
  3 -1 -1 -1 -1
```