Queue Carousel

You are managing a shop with a queue system for billing. Customers are initially arranged in a single line and need to be distributed across multiple queues. After distributing the customers into these queues, you need to process them in a specific order. The goal is to determine and print the sequence in which customers are dequeued from these queues until all queues are empty.

Details

Queue Initialization:

- Distribute the m customers into n queues. The distribution should be done such that each queue gets approximately m / n customers. If m is not perfectly divisible by n, the last queue may contain fewer customers.

Dequeuing Process:

- Process the queues in a round-robin fashion:

1. Start with the first queue and remove one customer at a time.
2. Move to the next queue in a round-robin manner, continuing until all queues are empty.
3. Repeat this process until there are no more customers left in any queue.

Input Format

Total Number of People (m): An integer m representing the total number of customers in the shop.
Number of Queues (n): An integer n representing the number of queues available for billing.
People Array (arr): An array of size m where each element is a unique integer representing a customer waiting in the shop.

Constraints

1 <= n <= m
All customers in the array are distinct

Output Format

Print a single line containing the order in which customers exit the queues. The output should be a space-separated list of customer identifiers.

Sample Input 0

```
6
2
1 2 3 4 5 6
```

Sample Output 0

```
1 4 2 5 3 6
```

Explanation 0

- The array [1, 2, 3, 4, 5, 6] is split into 2 queues:
  Queue 1: [1, 2, 3]
  Queue 2: [4, 5, 6]
- The queues are then dequeued in a round-robin fashion starting from the first queue:
  Dequeue from Queue 1: 1
  Dequeue from Queue 2: 4
  Dequeue from Queue 1: 2
  Dequeue from Queue 2: 5

Dequeue from Queue 1: 3
Dequeue from Queue 2: 6

- The final output is the sequence of dequeued elements: 1 4 2 5 3 6

Sample Input 1

```
6
3
1 2 3 4 5 6
```

Sample Output 1

```
1 3 5 2 4 6
```

Contest ends in an hour

Submissions: 31
Max Score: 10
Difficulty: Medium

Rate This Challenge:

More

C

```c
#include <stdio.h>
#include <stdlib.h>

#define MAX_QUEUE_SIZE 100

typedef struct {
  int data[MAX_QUEUE_SIZE];
  int front;
  int rear;
  int size;
} Queue;

void initializeQueue(Queue* queue) {
  queue->front = -1;
  queue->rear = -1;
  queue->size = 0;
}

void enqueue(Queue* queue, int person)
{
  if (queue->rear == MAX_QUEUE_SIZE-1) {
    printf("queue is full\n");
    return;
  }
  if (queue->rear == -1) {
    queue->front = 0;
  }

  (queue->rear)++;
  queue->data[queue->rear] = person;
}

int dequeue(Queue* queue)
{
  int data = queue->data[queue->front];
  (queue->front)++;
  return data;
}

int isQueueEmpty(Queue* queue)
```

```c
{
  return (queue->rear == -1 || queue->front - 1 == queue->rear);
}

Queue* createQueue(void)
{
  Queue* queue = (Queue*)malloc(sizeof(Queue));
  if (queue == NULL)
  {

    exit(EXIT_FAILURE);
  }
  initializeQueue(queue);
  return queue;

}

void roundRobinDequeue(Queue** queues, int n)
{
  int i = 0;
  int x = 0;
  while (1) {
    if (isQueueEmpty(queues[i])) {
      if (x == n)
        break;
      x++;
      i = (i + 1) % n;
      continue;
    }

    x = 0;
    int p = dequeue(queues[i]);
    printf("%d ", p);
    i = (i + 1) % n;
  }
}

int main() {
  int m, n;

  // Read the total number of people and number of queues
  scanf("%d", &m);
  scanf("%d", &n);

  // Read the people array
  int* arr = (int*)malloc(m * sizeof(int));
  for (int i = 0; i < m; ++i) {
    scanf("%d", &arr[i]);
  }

  // Create and initialize queues
  Queue** queues = (Queue**)malloc(n * sizeof(Queue*));
  for (int i = 0; i < n; ++i) {
    queues[i] = createQueue();
  }

  // Distribute people into queues
  int index = 0;
  for (int i = 0; i < m; ++i) {
    enqueue(queues[index], arr[i]);
    if ((i + 1) % (m / n) == 0 && index < n - 1) {
      index++;
    }
  }

  // Perform the round-robin dequeuing
  roundRobinDequeue(queues, n);

  // Free memory
  free(arr);
  for (int i = 0; i < n; ++i) {
    free(queues[i]);
  }
```

```
114     free(queues);
115
116     return 0;
117 }
```

Line: 117 Col: 2

Upload Code as File Test against custom input Run Code Submit Code

Testcase 0 ✔ Testcase 1 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
6
2
1 2 3 4 5 6
```

Your Output (stdout)

```
1 4 2 5 3 6
```

Expected Output

```
1 4 2 5 3 6
```