Bill Splitting

N people go to a restaurant. They sit around table such that the 1st person is next to the nth person. When the bill comes they play a game.They randomly select the number k and start counting from the 1st person and every kth person is eliminated i.e doesn't have to pay. The last three people remaining will split the bill. Write the code to find the last three people remaining using a circular linked list and for the helper function deleteNode.

Your task is to implement the helper function deleteNode and Josephus function: The deleteNode function should delete the current node from the CLL and adjusts the links accordingly void deleteNode(CircularLinkedList *list, Node *prev, Node *current); list:The circular linked list. prev:The node immediately before the current node. current:The node to be deleted.

Solve the Josephus problem by eliminating every k-th node until only 3 nodes remain void josephus(CircularLinkedList *list, int k); k:The step count(every k-th node will be removed)

Input Format

number_of_people(n)
Step_size(k)

Constraints

Note that the index starts from 1.
3<= number of people <=1000

Output Format

Indices of the last three people remaining.

Sample Input 0

```
7
3
```

Sample Output 0

```
1 4 5
```

Explanation 0

The counting starts from 1. The 3rd person is eliminated first. Then the 6th person is eliminated. Then the 2nd person is eliminated. Then the 7th person is eliminated leaving the 1st, 4th and 5th to pay the bill.

Contest ends in 9 hours

Submissions: 9
Max Score: 10
Difficulty: Medium

Rate This Challenge:

More

C

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

typedef struct Node {
    int data;
    struct Node *next;
} Node;

typedef struct {
    Node *head;
    Node *tail;
    int size;
} CircularLinkedList;

// Initialize the circular linked list to be empty
void initList(CircularLinkedList *list);
// Insert a node with the given value at the end of the circular linked list
void insertEnd(CircularLinkedList *list, int value);
// Solve the Josephus problem by eliminating every k-th node until only 3 nodes remain
void josephus(CircularLinkedList *list, int k);
// Print the information of all the nodes in the circular linked list
void printList(CircularLinkedList *list);
// Destroy the nodes in the circular linked list and reinitialize it to be empty
void destroyList(CircularLinkedList *list);
// Delete the current node from the circular linked list
void deleteNode(CircularLinkedList *list, Node *prev, Node *current);

int main() {
    int n, k;
    CircularLinkedList list;

    initList(&list);

    scanf("%d", &n);
    scanf("%d", &k);

    for (int i = 1; i <= n; i++) {
        insertEnd(&list, i);
    }

    josephus(&list, k);
    printList(&list);
    destroyList(&list);
    return 0;
}

void deleteNode(CircularLinkedList *list, Node *prev, Node *current) {
    if (list->head == list->tail && list->head == current) {
        // for only one node in the list
        free(current);
        list->head = list->tail = NULL;
    } else {
        if (current == list->head) {
            // If current node is head
            list->head = current->next;
        }
        if (current == list->tail) {
            // If current node is tail
            list->tail = prev;
        }
        prev->next = current->next; // skip the current node
        free(current);
    }
    list->size--;
}

void josephus(CircularLinkedList *list, int k) {
    Node *prev = list->tail;
    Node *current = list->head;

```

```c
    // Continue elimination until only 3 people remain
    while (list->size > 3) {
        // Move to the k-th person
        for (int i = 1; i < k; i++) {
            prev = current;
            current = current->next;
        }

        // Eliminate the k-th person
        deleteNode(list, prev, current);

        // Move the current pointer to the next person after elimination
        current = prev->next;
    }
}

void initList(CircularLinkedList *list) {
    list->head = NULL;
    list->tail = NULL;
    list->size = 0;
}

void insertEnd(CircularLinkedList *list, int value) {
    Node *newNode = (Node *)malloc(sizeof(Node));
    newNode->data = value;
    newNode->next = NULL;

    if (list->head == NULL) {
        list->head = newNode;
        list->tail = newNode;
        newNode->next = list->head;
    } else {
        list->tail->next = newNode;
        list->tail = newNode;
        list->tail->next = list->head; // Circular linking
    }
    list->size++;
}

void printList(CircularLinkedList *list) {
    if (list->head == NULL) {
        return;
    }

    Node *current = list->head;
    do {
        printf("%d ", current->data);
        current = current->next;
    } while (current != list->head);

    printf("\n");
}

void destroyList(CircularLinkedList *list) {
    if (list->head != NULL) {
        Node *p = list->head;

        while (p != list->tail) {
            list->head = p->next;
            list->tail->next = p->next;
            free(p);
            p = list->head;
        }
        free(p);
        list->head = NULL;
        list->tail = NULL;
    }
}
```

Line: 63 Col: 6

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

Testcase 0 ✔

Congratulations, you passed the sample test case.

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

Input (stdin)

```
7
3
```

Your Output (stdout)

```
1 4 5
```

Expected Output

```
1 4 5
```

Interview Prep | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy |