Name : Shrinivas Hatyalikar
Div: TY-B (B2)
Roll No: 26

Q) Josephus Problem

Part 1 : Array Implementation

Code:

```c
#include <stdio.h>

int josephus(int n, int k) {
    // Create an array to represent the circle of people
    int circle[n];
    for (int i = 0; i < n; i++) {
        circle[i] = i + 1;
    }

    // Initialize the starting index (position) as 0
    int index = 0;

    // Iterate until there's only one person left in the circle
    while (n > 1) {
        // Calculate the index of the person to be removed
        index = (index + k - 1) % n;

        // Remove the person from the circle by shifting elements
        for (int i = index; i < n - 1; i++) {
            circle[i] = circle[i + 1];
        }

        // Decrement the number of people in the circle
        n--;
    }

    // Return the position of the last remaining person
    return circle[0];
}

int main() {
    int n = 0;  // Number of people in the circle
    int k = 2;  // Step size to count (the k-th person will be removed each time)
    printf("Enter the number of People: ");
```

```c
    scanf("%d",&n);
    int result = josephus(n, k);
    printf("The last remaining person's position is: %d\n", result);

    return 0;
}
```

Output:

```
PS C:\Users\sheeh\OneDrive\Desktop\C\output> & .\'_3rdlab.exe'
Enter the number of People: 5
The last remaining person's position is: 3
PS C:\Users\sheeh\OneDrive\Desktop\C\output>
```

Part 2 : Circular Queue Implementation

Code:

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

struct Node {
    int value;
    struct Node* next;
};

struct Node* createNewNode(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->value = value;
    newNode->next = NULL;
    return newNode;
}

void addToQueue(struct Node** rear, int value) {
    struct Node* newNode = createNewNode(value);
    if (*rear == NULL) {
        *rear = newNode;
        (*rear)->next = newNode;
    } else {
        newNode->next = (*rear)->next;
        (*rear)->next = newNode;
```

```
        *rear = newNode;
    }
}

int removeFromQueue(struct Node** rear) {
    if (*rear == NULL)
        return -1;

    struct Node* front = (*rear)->next;
    int value = front->value;
    if (front == *rear) {
        free(front);
        *rear = NULL;
    } else {
        (*rear)->next = front->next;
        free(front);
    }
    return value;
}

int findWinningPerson(int n) {
    struct Node* rear = NULL;

    for (int i = 1; i <= n; i++) {
        addToQueue(&rear, i);
    }

    while (rear->next != rear) {
        int firstPerson = removeFromQueue(&rear);
        int secondPerson = removeFromQueue(&rear);
        addToQueue(&rear, firstPerson);
    }

    int winningPerson = rear->value;

    free(rear);

    return winningPerson;
}

int main() {
    int n = 0;
    printf("Enter number of persons: ");
    scanf("%d",&n);

    int winner = findWinningPerson(n);
    printf("Using Circular Queue\n");
    printf("The last remaining person's position is: %d", winner);
```

```
    return 0;
}
```

Output:

```
Enter number of persons: 5
Using Circular Queue
The last remaining person's position is: 3
PS C:\Users\sheeh\OneDrive\Desktop\C\output>
```

Part 3 : Recursion Implementation

Recurrance Relation:

f(n) = Winners position when the total no. of people = n

f(n) -> 2 f(j) – 1 n:even , n=2j

-> 2 f(j) + 1 n:odd , n=2j+1

Code:

```c
#include <stdio.h>

int remain[1000]; // Assuming n <= 1000, adjust the array size as needed

int findremaining(int n) {
    if (n == 1) {
        return 1;
    }

    if (remain[n] != -1) {
        return remain[n];
    }

    int j;
    if (n % 2 == 0) {
        j = n / 2;
        remain[n] = 2 * findremaining(j) - 1;
    } else {
        j = (n - 1) / 2;
        remain[n] = 2 * findremaining(j) + 1;
    }

    return remain[n];
```

```c
}

int main() {
    int n;
    printf("Enter number of persons: ");
    scanf("%d", &n);

    for (int i = 0; i <= n; i++) {
        remain[i] = -1; // Initialize the memoization array
    }

    int remain = findremaining(n);
    printf("Using Recurssion\n");
    printf("The last remaining person's position is: %d\n", remain);

    return 0;
}
```

Output:

```
Enter number of persons: 5
Using Recurssion
The last remaining person's position is: 3
PS C:\Users\sheeh\OneDrive\Desktop\C\output>
```