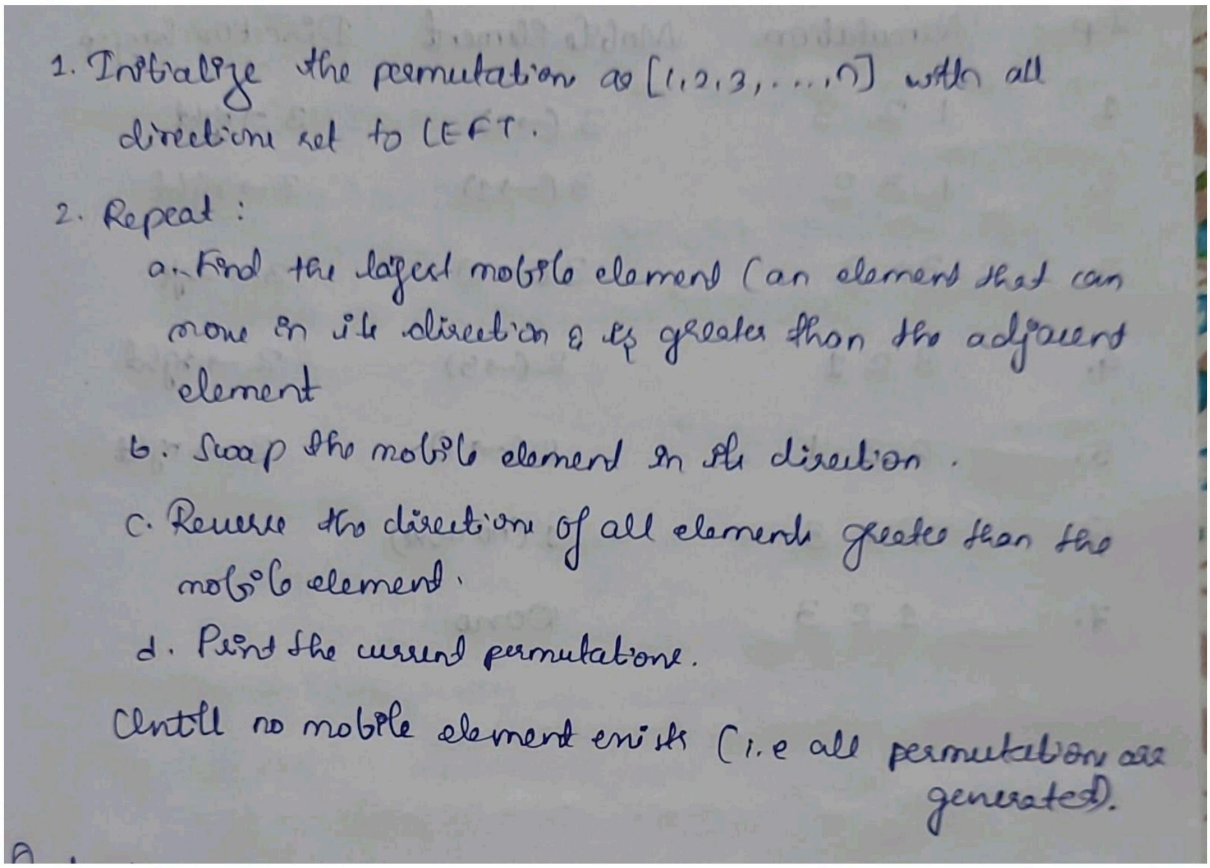
PROGRAM 10

Implement Johnson Trotter algorithm to generate permutations.

ALGORITHM


1. Initialize the permutation as [1,2,3,...,n] with all directions set to LEFT.

2. Repeat :
   a. Find the largest mobile element (an element that can move in its direction & is greater than the adjacent element
   b. Swap the mobile element in its direction.
   c. Reverse the directions of all elements greater than the mobile element.
   d. Print the current permutations.

Until no mobile element exists (i.e all permutations are generated).

CODE

```
#include <stdio.h>

#define MAX 20
#define LEFT -1
#define RIGHT 1

int n;
int perm[MAX];     // Current permutation
int dir[MAX];      // Directions: -1 for LEFT, 1 for RIGHT
```

```
// Function to print the current permutation
void printPerm() {
    for (int i = 0; i < n; i++)
        printf("%d ", perm[i]);
    printf("\n");
}

// Function to find the largest mobile integer
int getMobile() {
    int mobile = 0, mobileIndex = -1;

    for (int i = 0; i < n; i++) {
        int next = i + dir[i];
        if (next >= 0 && next < n && perm[i] > perm[next]) {
            if (perm[i] > mobile) {
                mobile = perm[i];
                mobileIndex = i;
            }
        }
    }

    return mobileIndex;
}

// Function to swap two elements and their directions
void swap(int i, int j) {
    int temp = perm[i];
    perm[i] = perm[j];
    perm[j] = temp;
```

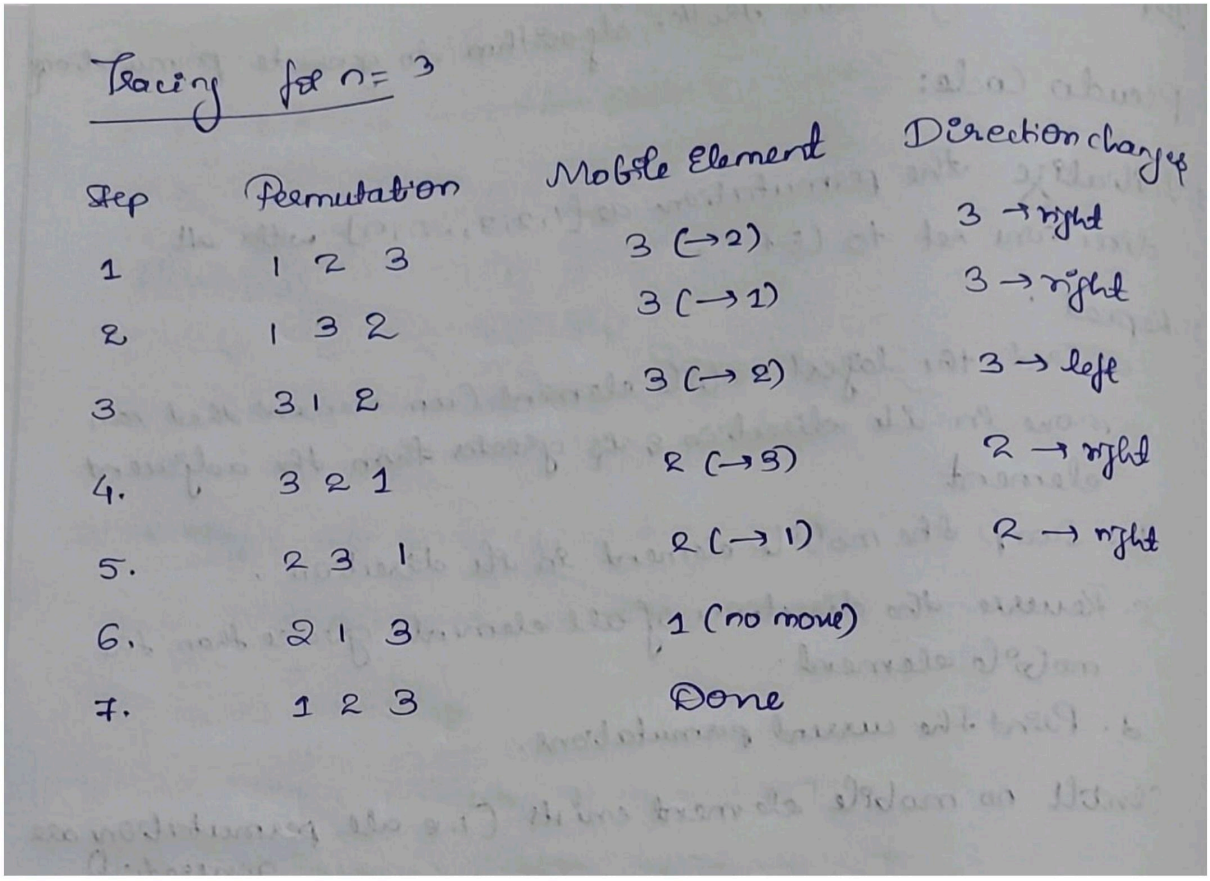
```
    int dtemp = dir[i];
    dir[i] = dir[j];
    dir[j] = dtemp;
}

void generatePermutations() {
    printPerm();  // First permutation

    while (1) {
        int mobileIndex = getMobile();
        if (mobileIndex == -1)
            break;

        int next = mobileIndex + dir[mobileIndex];
        swap(mobileIndex, next);

        // After moving, change direction of all elements > moved element
        for (int i = 0; i < n; i++) {
            if (perm[i] > perm[next])
                dir[i] *= -1;
        }

        printPerm();
    }
}

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

```
    // Initialize permutation and directions
    for (int i = 0; i < n; i++) {
        perm[i] = i + 1;
        dir[i] = LEFT;
    }

    printf("All permutations using Johnson-Trotter:\n");
    generatePermutations();

    return 0;
}
```

OUTPUT

```
Enter number of elements (n): 3
All permutations using Johnson-Trotter:
1 2 3
1 3 2
3 1 2
3 2 1
2 3 1
2 1 3
```

TRACING



Tracing for n = 3

Step	Permutation	Mobile Element	Direction change
1	1 2 3	3 (→ 2)	3 → right
2	1 3 2	3 (→ 1)	3 → right
3	3 1 2	3 (→ 2)	3 → left
4.	3 2 1	2 (→ 3)	2 → right
5.	2 3 1	2 (→ 1)	2 → right
6.	2 1 3	1 (no move)	
7.	1 2 3	Done	