

Week 07:

Johnson Trotter program :

CODE:-

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

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

```
#include <stdbool.h>
```

```
void printPermutation(int *perm, int size) {
```

```
    for (int i = 0; i < size; i++) {
```

```
        printf("%d ", perm[i]);
```

```
    }
```

```
    printf("\n");
```

```
}
```

```
int findLargestMobile(int *perm, int *dir, int size) {
```

```
    int largestMobileIndex = -1;
```

```
    int largestMobileValue = -1;
```

```
    for (int i = 0; i < size; i++) {
```

```
        int nextIndex = i + dir[i];
```

```
        if (nextIndex >= 0 && nextIndex < size && perm[i] > perm[nextIndex] && perm[i] > largestMobileValue) {
```

```
            largestMobileValue = perm[i];
```

```
            largestMobileIndex = i;
```

```
        }
```

```
    }
```

```
        return largestMobileIndex;
    }
}
```

```
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

```
void generatePermutations(int n) {
    int *perm = (int *)malloc(n * sizeof(int));
    int *dir = (int *)malloc(n * sizeof(int));

    for (int i = 0; i < n; i++) {
        perm[i] = i + 1;
        dir[i] = -1; // Initialize all directions to left (-1)
    }

    printPermutation(perm, n);

    while (true) {
        int largestMobileIndex = findLargestMobile(perm, dir, n);
        if (largestMobileIndex == -1) {
            break; // No more mobile integers
        }
    }
}
```

```
}
```

```
int nextIndex = largestMobileIndex + dir[largestMobileIndex];
```

```
swap(&perm[largestMobileIndex], &perm[nextIndex]);
```

```
swap(&dir[largestMobileIndex], &dir[nextIndex]);
```

```
for (int i = 0; i < n; i++) {
```

```
    if (perm[i] > perm[nextIndex]) {
```

```
        dir[i] = -dir[i]; // Reverse the direction
```

```
    }
```

```
}
```

```
printPermutation(perm, n);
```

```
}
```

```
free(perm);
```

```
free(dir);
```

```
}
```

```
int main() {
```

```
    int n;
```

```
    printf("Enter the number of elements: ");
```

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

```
    generatePermutations(n);
```

```
        return 0;
    }
}
```

OUTPUT:-

```
Johnson Trotter algorithm to find all permutations of given numbers
Enter the number
3
Total permutations = 6
All possible permutations are:
1 2 3
1 3 2
3 1 2
3 2 1
2 3 1
2 1 3
```

LEETCODE

To find the Kth largest integer in the given array

You are given an array of strings `nums` and an integer `k`. Each string in `nums` represents an integer without leading zeros.

Return *the string that represents the k^{th} **largest integer** in `nums`.*

code:-

```
int compare(const void *a, const void *b) {
    const char *str1 = *(const char **)a;
    const char *str2 = *(const char **)b;

    int len1 = strlen(str1);
    int len2 = strlen(str2);
    if (len1 != len2) {
        return len2 - len1;
    }
}
```

```

        return strcmp(str2, str1);
    }
}

```

```

char* kthLargestNumber(char **nums, int numsSize, int k) {
    qsort(nums, numsSize, sizeof(char*), compare);
    return nums[k - 1];
}

```

Ouput:-

✓ Testcase | >_ Test Result

Accepted Runtime: 2 ms

• Case 1 • Case 2 • Case 3

Input

nums =
["3", "6", "7", "10"]

k =
4

Output

"3"

Expected

"3"

✓ Testcase | >_ Test Result

Accepted Runtime: 2 ms

• Case 1 • **Case 2** • Case 3

Input

nums =
["2", "21", "12", "1"]

k =
3

Output

"2"

Expected

"2"

✓ Testcase | >_ Test Result

Accepted Runtime: 2 ms

• Case 1 • Case 2 • **Case 3**

Input

nums =
["0", "0"]

k =
2

Output

"0"

Expected

"0"

Substring matching program in main Text program :brute force technique

CODE:-

```
#include <stdio.h>

#include <string.h>

int substringMatch(char *text, char *pattern) {

    int textLength = strlen(text);

    int patternLength = strlen(pattern);

    for (int i = 0; i <= textLength - patternLength; i++) {

        int j;

        for (j = 0; j < patternLength; j++) {

            if (text[i + j] != pattern[j])

                break;

        }

        if (j == patternLength)

            return i;

    }

    return -1;

}

int main() {

    char text[100], pattern[100];

    printf("Enter the main text: ");

    fgets(text, sizeof(text), stdin);
```

```
text[strcspn(text, "\n")] = '\0';

printf("Enter the pattern to search: ");

fgets(pattern, sizeof(pattern), stdin);

pattern[strcspn(pattern, "\n")] = '\0';

int index = substringMatch(text, pattern);

if (index != -1)

    printf("Pattern found at index: %d\n", index);

else

    printf("Pattern not found in the text.\n");

return 0;

}
```

OUTPUT:-

```
Enter the main text: hello welcome to bmsce
Enter the pattern to search: welcome
Pattern found at index: 6

Process returned 0 (0x0)   execution time : 33.046 s
Press any key to continue.
```