Ice Cream Parlor

```
#include <assert.h>
#include <ctype.h>
#include <limits.h>
#include <math.h>
#include <stdbool.h>
#include <stddef.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char* readline();
char* ltrim(char*);
char* rtrim(char*);
char** split_string(char*);

int parse_int(char*);

/*
 * Complete the 'icecreamParlor' function below.
 *
 * The function is expected to return an INTEGER_ARRAY.
 * The function accepts following parameters:
 *  1. INTEGER m
 *  2. INTEGER_ARRAY arr
 */

/*
 * To return the integer array from the function, you should:
 *     - Store the size of the array to be returned in the
result_count variable
 *     - Allocate the array statically or dynamically
 *
 * For example,
 * int* return_integer_array_using_static_allocation(int*
result_count) {
 *     *result_count = 5;
 *
 *     static int a[5] = {1, 2, 3, 4, 5};
 *
 *     return a;
 * }
 *
```

```
 * int* return_integer_array_using_dynamic_allocation(int*
result_count) {
 *     *result_count = 5;
 *
 *     int *a = malloc(5 * sizeof(int));
 *
 *     for (int i = 0; i < 5; i++) {
 *         *(a + i) = i + 1;
 *     }
 *
 *     return a;
 * }
 *
 */
int* icecreamParlor(int m, int arr_count, int* arr, int*
result_count) {
    *result_count=2;
    int* res=malloc(2*sizeof(int));
    for(int i=0;i<arr_count;i++){
        for(int j=i+1;j<arr_count;j++){
            if(arr[i]+arr[j]==m){
                res[0]=i+1;
                res[1]=j+1;
                return res;
            }
        }
    }
    return res;
}

int main()
{
    FILE* fptr = fopen(getenv("OUTPUT_PATH"), "w");

    int t = parse_int(ltrim(rtrim(readline())));

    for (int t itr = 0; t itr < t; t itr++) {
        int m = parse int(ltrim(rtrim(readline())));

        int n = parse int(ltrim(rtrim(readline())));

        char** arr_temp = split_string(rtrim(readline()));

        int* arr = malloc(n * sizeof(int));

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

```
            int arr_item = parse_int(*(arr_temp + i));

            *(arr + i) = arr_item;
        }

        int result_count;
        int* result = icecreamParlor(m, n, arr, &result_count);

        for (int i = 0; i < result_count; i++) {
            fprintf(fptr, "%d", *(result + i));

            if (i != result_count - 1) {
                fprintf(fptr, " ");
            }
        }

        fprintf(fptr, "\n");
    }

    fclose(fptr);

    return 0;
}

char* readline() {
    size_t alloc_length = 1024;
    size_t data_length = 0;

    char* data = malloc(alloc_length);

    while (true) {
        char* cursor = data + data_length;
        char* line = fgets(cursor, alloc_length - data_length,
stdin);

        if (!line) {
            break;
        }

        data_length += strlen(cursor);

        if (data_length < alloc_length - 1 || data[data_length -
1] == '\n') {
            break;
        }
```

```
            alloc_length <<= 1;

            data = realloc(data, alloc_length);

            if (!data) {
                data = '\0';

                break;
            }
        }

        if (data[data_length - 1] == '\n') {
            data[data_length - 1] = '\0';

            data = realloc(data, data_length);

            if (!data) {
                data = '\0';
            }
        } else {
            data = realloc(data, data_length + 1);

            if (!data) {
                data = '\0';
            } else {
                data[data_length] = '\0';
            }
        }

        return data;
    }

    char* ltrim(char* str) {
        if (!str) {
            return '\0';
        }

        if (!*str) {
            return str;
        }

        while (*str != '\0' && isspace(*str)) {
            str++;
        }

        return str;
```

```
}

char* rtrim(char* str) {
    if (!str) {
        return '\0';
    }

    if (!*str) {
        return str;
    }

    char* end = str + strlen(str) - 1;

    while (end >= str && isspace(*end)) {
        end--;
    }

    *(end + 1) = '\0';

    return str;
}

char** split_string(char* str) {
    char** splits = NULL;
    char* token = strtok(str, " ");

    int spaces = 0;

    while (token) {
        splits = realloc(splits, sizeof(char*) * ++spaces);

        if (!splits) {
            return splits;
        }

        splits[spaces - 1] = token;

        token = strtok(NULL, " ");
    }

    return splits;
}

int parse_int(char* str) {
    char* endptr;
    int value = strtol(str, &endptr, 10);
```

```
    if (endptr == str || *endptr != '\0') {
        exit(EXIT_FAILURE);
    }

    return value;
}
```