Waiter

```
#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 'waiter' function below.
 *
 * The function is expected to return an INTEGER_ARRAY.
 * The function accepts following parameters:
 *  1. INTEGER_ARRAY number
 *  2. INTEGER q
 */

/*
 * 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 prime(int n){
    int count=0;
    for(int i=2;;i++){
        int f=1;
        for(int j=2;j*j<=i;j++){
            if(i%j==0){f=0;break;}
        }
        if(f)count++;
        if(count==n)return i;
    }
}

int* waiter(int number_count, int* number, int q, int*
result_count) {
    int* ans=malloc(number_count*sizeof(int));
    int top=0;
    int* A=malloc(number_count*sizeof(int));
    int Asize=number_count;
    for(int i=0;i<number_count;i++)A[i]=number[i];
    for(int i=1;i<=q;i++){
        int p=prime(i);
        int* B=malloc(Asize*sizeof(int));
        int* nextA=malloc(Asize*sizeof(int));
        int Btop=0,Ntop=0;
        for(int j=Asize-1;j>=0;j--){
            if(A[j]%p==0)B[Btop++]=A[j];
            else nextA[Ntop++]=A[j];
        }
        for(int j=Btop-1;j>=0;j--)ans[top++]=B[j];
        free(A);
        A=nextA;
```

```
        Asize=Ntop;
        free(B);
    }
    for(int j=Asize-1;j>=0;j--)ans[top++]=A[j];
    free(A);
    *result_count=top;
    return ans;
}

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

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

    int n = parse_int(*(first_multiple_input + 0));

    int q = parse_int(*(first_multiple_input + 1));

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

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

    for (int i = 0; i < n; i++) {
        int number_item = parse_int(*(number_temp + i));

        *(number + i) = number_item;
    }

    int result_count;
    int* result = waiter(n, number, q, &result_count);

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

        if (i != result count - 1) {
            fprintf(fptr, "\n");
        }
    }

    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;
}
```