```
1. #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 'reverseArray' function below.
 *
 * The function is expected to return an INTEGER_ARRAY.
 * The function accepts INTEGER_ARRAY a as parameter.
 */
int* reverseArray(int a_count, int* a, int* result_count) {
    *result_count = a_count;                        // set size of
returned array
    int *result = malloc(a_count * sizeof(int));    // allocate result
array

    for (int i = 0; i < a_count; i++) {
        result[i] = a[a_count - 1 - i];             // reverse the array
    }

    return result;
}

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

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

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

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

    for (int i = 0; i < arr_count; i++) {
        int arr_item = parse_int(*(arr_temp + i));
        *(arr + i) = arr_item;
    }

    int res_count;
    int* res = reverseArray(arr_count, arr, &res_count);

    for (int i = 0; i < res_count; i++) {
        fprintf(fptr, "%d", *(res + i));
        if (i != res_count - 1) {
```

```
            fprintf(fptr, " ");
        }
    }

    fprintf(fptr, "\n");

    fclose(fptr);

    return 0;
}

/* ---- Helper functions below ---- */

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;
}
```

2.

```
#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 'hourglassSum' function below.
 * The function is expected to return an INTEGER.
 * The function accepts 2D_INTEGER_ARRAY arr as parameter.
 */
int hourglassSum(int arr_rows, int arr_columns, int** arr) {
    int maxSum = INT_MIN;

    for (int i = 0; i <= arr_rows - 3; i++) {
        for (int j = 0; j <= arr_columns - 3; j++) {

            int sum =
                arr[i][j] + arr[i][j+1] + arr[i][j+2] +
                            arr[i+1][j+1] +
                arr[i+2][j] + arr[i+2][j+1] + arr[i+2][j+2];
```

```
            if (sum > maxSum)
                maxSum = sum;
        }
    }

    return maxSum;
}

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

    int** arr = malloc(6 * sizeof(int*));

    for (int i = 0; i < 6; i++) {
        *(arr + i) = malloc(6 * sizeof(int));

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

        for (int j = 0; j < 6; j++) {
            int arr_item = parse_int(*(arr_item_temp + j));
            *(*(arr + i) + j) = arr_item;
        }
    }

    int result = hourglassSum(6, 6, arr);

    fprintf(fptr, "%d\n", result);

    fclose(fptr);

    return 0;
}

/* ----------------------------------------
   Helper Functions
   ---------------------------------------- */

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;
}
```

3.
```
#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 'dynamicArray' function below.
 */
int* dynamicArray(int n, int queries_rows, int queries_columns, int**
queries, int* result_count) {

    int** seqList = malloc(n * sizeof(int*));
    int* seqSizes = malloc(n * sizeof(int));

    for (int i = 0; i < n; i++) {
        seqList[i] = NULL;
        seqSizes[i] = 0;
    }

    int lastAnswer = 0;

    int* results = malloc(queries_rows * sizeof(int));
    int res_idx = 0;

    for (int i = 0; i < queries_rows; i++) {

        int type = queries[i][0];
        int x = queries[i][1];
        int y = queries[i][2];

        int idx = (x ^ lastAnswer) % n;

        if (type == 1) {
```

```
            seqSizes[idx]++;
            seqList[idx] = realloc(seqList[idx], seqSizes[idx] *
sizeof(int));
            seqList[idx][seqSizes[idx] - 1] = y;

        } else if (type == 2) {
            int pos = y % seqSizes[idx];
            lastAnswer = seqList[idx][pos];
            results[res_idx++] = lastAnswer;
        }
    }

    *result_count = res_idx;
    return results;
}

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));

    int** queries = malloc(q * sizeof(int*));

    for (int i = 0; i < q; i++) {
        *(queries + i) = malloc(3 * sizeof(int));

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

        for (int j = 0; j < 3; j++) {
            int queries_item = parse_int(*(queries_item_temp + j));
            *(*(queries + i) + j) = queries_item;
        }
    }

    int result_count;
    int* result = dynamicArray(n, q, 3, queries, &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;
}


4. #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*);

int* rotateLeft(int d, int arr_count, int* arr, int* result_count) {

    *result_count = arr_count;
    int* result = malloc(arr_count * sizeof(int));

    d = d % arr_count;   // handle large rotations

    for (int i = 0; i < arr_count; i++) {
        result[i] = arr[(i + d) % arr_count];
    }

    return result;
}
```

```
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 d = parse_int(*(first_multiple_input + 1));

    char** arr_temp = split_string(rtrim(readline()));
    int* arr = malloc(n * sizeof(int));

    for (int i = 0; i < n; i++) {
        arr[i] = parse_int(arr_temp[i]);
    }

    int result_count;
    int* result = rotateLeft(d, 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 (1) {
        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) break;
    }

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

```
    return data;
}

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

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

    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;
    int spaces = 0;
    char* token = strtok(str, " ");

    while (token) {
        splits = realloc(splits, sizeof(char*) * ++spaces);
        splits[spaces - 1] = token;
        token = strtok(NULL, " ");
    }

    return splits;
}

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

```
5. #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*);

int parse_int(char*);
```

```
/*
 * Complete the 'matchingStrings' function below.
 *
 * The function is expected to return an INTEGER_ARRAY.
 * The function accepts following parameters:
 *  1. STRING_ARRAY stringList
 *  2. STRING_ARRAY queries
 */
int* matchingStrings(int stringList_count, char** stringList,
                     int queries_count, char** queries, int*
result_count) {

    *result_count = queries_count;
    int* result = malloc(queries_count * sizeof(int));

    for (int i = 0; i < queries_count; i++) {
        int count = 0;

        for (int j = 0; j < stringList_count; j++) {
            if (strcmp(queries[i], stringList[j]) == 0) {
                count++;
            }
        }

        result[i] = count;
    }

    return result;
}

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

    int stringList_count = parse_int(ltrim(rtrim(readline())));
    char** stringList = malloc(stringList_count * sizeof(char*));

    for (int i = 0; i < stringList_count; i++) {
        char* stringList_item = readline();
        stringList[i] = stringList_item;
    }

    int queries_count = parse_int(ltrim(rtrim(readline())));
    char** queries = malloc(queries_count * sizeof(char*));

    for (int i = 0; i < queries_count; i++) {
        char* queries_item = readline();
        queries[i] = queries_item;
    }

    int res_count;
    int* res = matchingStrings(stringList_count, stringList,
queries_count, queries, &res_count);

    for (int i = 0; i < res_count; i++) {
        fprintf(fptr, "%d", res[i]);
        if (i != res_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)
            return '\0';
    }

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

    return data;
}

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

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

    char* end = str + strlen(str) - 1;
    while (end >= str && isspace(*end))
        end--;

    *(end + 1) = '\0';
    return str;
```

```
}

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