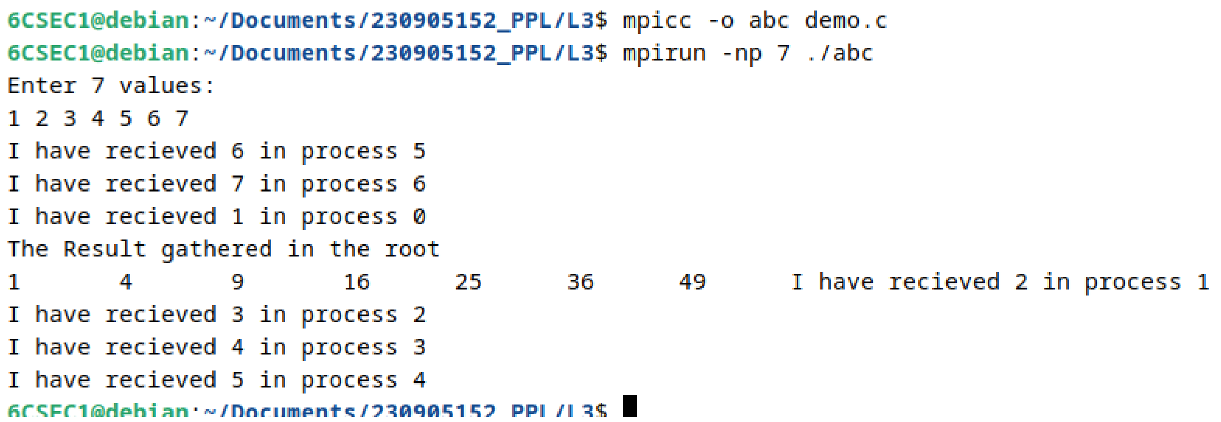
Lab 3- Collective Communication in MPI

Solved:

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

int main(int argc, char *argv[]) {
    int rank, size, N, A[10], B[10], i, c;

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    if (rank == 0) {
        N = size;
        fprintf(stdout, "Enter %d values: \n", N);
        fflush(stdout);
        for (int i = 0; i < N; ++i) {
            scanf("%d", &A[i]);
        }
    }

    MPI_Scatter(A, 1, MPI_INT, &c, 1, MPI_INT, 0, MPI_COMM_WORLD);

    fprintf(stdout, "I have received %d in process %d\n", c, rank);
    fflush(stdout);

    c = c * c;

    MPI_Gather(&c, 1, MPI_INT, B, 1, MPI_INT, 0, MPI_COMM_WORLD);

    if (rank == 0) {
        fprintf(stdout, "The Result gathered in the root \n");
        fflush(stdout);
        for (i = 0; i < N; ++i) {
            fprintf(stdout, "%d\t", B[i]);
            fflush(stdout);
        }
    }

    MPI_Finalize();
    return 0;
}
```

Output:

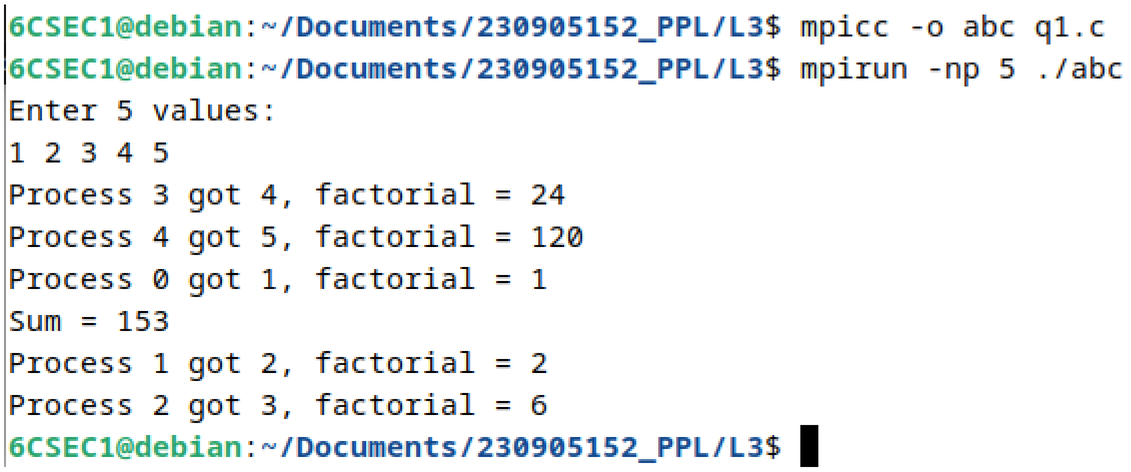


```
6CSEC1@debian:~/Documents/230905152_PPL/L3$ mpicc -o abc demo.c
6CSEC1@debian:~/Documents/230905152_PPL/L3$ mpirun -np 7 ./abc
Enter 7 values:
1 2 3 4 5 6 7
I have recieved 6 in process 5
I have recieved 7 in process 6
I have recieved 1 in process 0
The Result gathered in the root
1       4       9       16      25      36      49      I have recieved 2 in process 1
I have recieved 3 in process 2
I have recieved 4 in process 3
I have recieved 5 in process 4
6CSEC1@debian:~/Documents/230905152_PPL/L3$
```

Q1.

```c
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>

long long fact(int num) {
    long long fact = 1;
    for (int i = 1; i <= num; i++) fact *= i;
    return fact;
}

int main(int argc, char *argv[]) {
    int rank, size;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    int *numbers = NULL, num;
    long long fact, sum;

    if (rank == 0) {
        numbers = malloc(size * sizeof(int));
        fprintf(stdout, "Enter %d values: \n", size);
        fflush(stdout);
        for (int i = 0; i < size; i++) scanf("%d", &numbers[i]);
    }

    MPI_Scatter(numbers, 1, MPI_INT, &num, 1, MPI_INT, 0, MPI_COMM_WORLD);
    fact = fact(num);

    long long *results = NULL;
    if (rank == 0) results = malloc(size * sizeof(long long));
    MPI_Gather(&fact, 1, MPI_LONG_LONG, results, 1, MPI_LONG_LONG, 0,
MPI_COMM_WORLD);

    for (int i = 0; i < size; i++) {
        MPI_Barrier(MPI_COMM_WORLD);
        if (rank == i) {
            printf("Process %d got %d, factorial = %lld\n", rank, num, fact);
            fflush(stdout);
        }
    }

    if (rank == 0) {
        sum = 0;
        for (int i = 0; i < size; i++) sum += results[i];
        printf("Sum = %lld\n", sum);
        free(numbers);
        free(results);
    }

    MPI_Finalize();
    return 0;
}
```

Output:

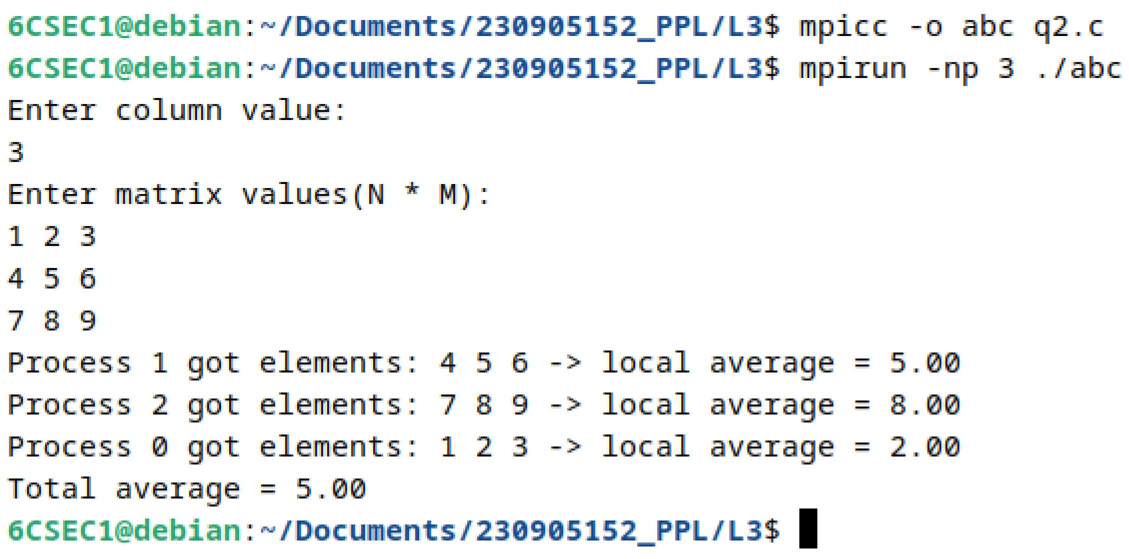


```
6CSEC1@debian:~/Documents/230905152_PPL/L3$ mpicc -o abc q1.c
6CSEC1@debian:~/Documents/230905152_PPL/L3$ mpirun -np 5 ./abc
Enter 5 values:
1 2 3 4 5
Process 3 got 4, factorial = 24
Process 4 got 5, factorial = 120
Process 0 got 1, factorial = 1
Sum = 153
Process 1 got 2, factorial = 2
Process 2 got 3, factorial = 6
6CSEC1@debian:~/Documents/230905152_PPL/L3$ 
```

Q2.

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

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

int main(int argc, char *argv[]) {
    int rank, size;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    int M;
    int *array = NULL;
    int *subarray;
    double avg, total_avg;

    if (rank == 0) {
        fprintf(stdout, "Enter column value: \n");
        fflush(stdout);
        scanf("%d", &M);
        array = malloc(size * M * sizeof(int));
        fprintf(stdout, "Enter matrix values(N * M): \n");
        fflush(stdout);
        for (int i = 0; i < size * M; i++) {
            scanf("%d", &array[i]);
        }
    }

    MPI_Bcast(&M, 1, MPI_INT, 0, MPI_COMM_WORLD);

    subarray = malloc(M * sizeof(int));
    MPI_Scatter(array, M, MPI_INT, subarray, M, MPI_INT, 0, MPI_COMM_WORLD);

    int sum = 0;
    for (int i = 0; i < M; i++) sum += subarray[i];
    avg = (double)sum / M;

    double *all_avgs = NULL;
    if (rank == 0) {
        all_avgs = malloc(size * sizeof(double));
    }
    MPI_Gather(&avg, 1, MPI_DOUBLE, all_avgs, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);

    for (int i = 0; i < size; i++) {
        MPI_Barrier(MPI_COMM_WORLD);
        if (rank == i) {
            printf("Process %d got elements: ", rank);
            for (int j = 0; j < M; j++) printf("%d ", subarray[j]);
            printf("-> local average = %.2f\n", avg);
            fflush(stdout);
        }
    }

    if (rank == 0) {
        double sum_avgs = 0.0;
        for (int i = 0; i < size; i++) sum_avgs += all_avgs[i];
        total_avg = sum_avgs / size;
        printf("Total average = %.2f\n", total_avg);
        free(array);
        free(all_avgs);
    }

    free(subarray);
    MPI_Finalize();
    return 0;
}
```
Output:



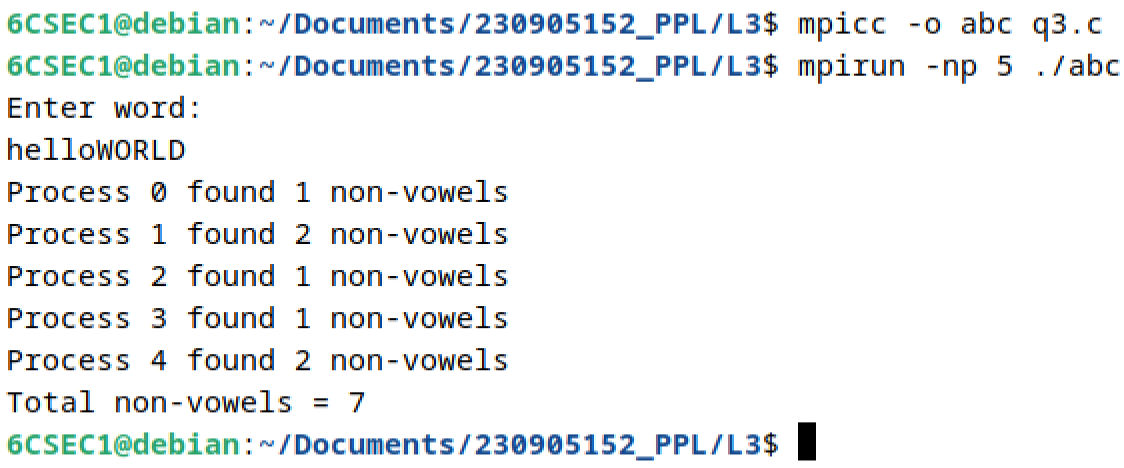
```
6CSEC1@debian:~/Documents/230905152_PPL/L3$ mpicc -o abc q2.c
6CSEC1@debian:~/Documents/230905152_PPL/L3$ mpirun -np 3 ./abc
Enter column value:
3
Enter matrix values(N * M):
1 2 3
4 5 6
7 8 9
Process 1 got elements: 4 5 6 -> local average = 5.00
Process 2 got elements: 7 8 9 -> local average = 8.00
Process 0 got elements: 1 2 3 -> local average = 2.00
Total average = 5.00
6CSEC1@debian:~/Documents/230905152_PPL/L3$ 
```

Q3.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mpi.h>
#include <ctype.h>
```

```c
int is_vowel(char c) {
    c = tolower(c);
    return (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u');
}

int main(int argc, char *argv[]) {
    int rank, size;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    char *str = NULL;
    int len, chunk;
    char *sub;
    int local_count = 0;
    int *counts = NULL;

    if (rank == 0) {
        char buffer[1024];
        fprintf(stdout, "Enter word: \n");
        fflush(stdout);
        scanf("%s", buffer);
        len = strlen(buffer);
        if (len % size != 0) {
            MPI_Abort(MPI_COMM_WORLD, 1);
        }
        str = buffer;
    }

    MPI_Bcast(&len, 1, MPI_INT, 0, MPI_COMM_WORLD);
    chunk = len / size;
    sub = (char *)malloc(chunk);

    MPI_Scatter(str, chunk, MPI_CHAR, sub, chunk, MPI_CHAR, 0, MPI_COMM_WORLD);

    for (int i = 0; i < chunk; i++) {
        if (!is_vowel(sub[i])) {
            local_count++;
        }
    }

    if (rank == 0) {
        counts = (int *)malloc(size * sizeof(int));
    }
    MPI_Gather(&local_count, 1, MPI_INT, counts, 1, MPI_INT, 0, MPI_COMM_WORLD);

    if (rank == 0) {
        int total = 0;
        for (int i = 0; i < size; i++) {
            printf("Process %d found %d non-vowels\n", i, counts[i]);
            total += counts[i];
        }
        printf("Total non-vowels = %d\n", total);
        free(counts);
    }

    free(sub);
    MPI_Finalize();
    return 0;
}
```

Output:



```
6CSEC1@debian:~/Documents/230905152_PPL/L3$ mpicc -o abc q3.c
6CSEC1@debian:~/Documents/230905152_PPL/L3$ mpirun -np 5 ./abc
Enter word:
helloWORLD
Process 0 found 1 non-vowels
Process 1 found 2 non-vowels
Process 2 found 1 non-vowels
Process 3 found 1 non-vowels
Process 4 found 2 non-vowels
Total non-vowels = 7
6CSEC1@debian:~/Documents/230905152_PPL/L3$ 
```

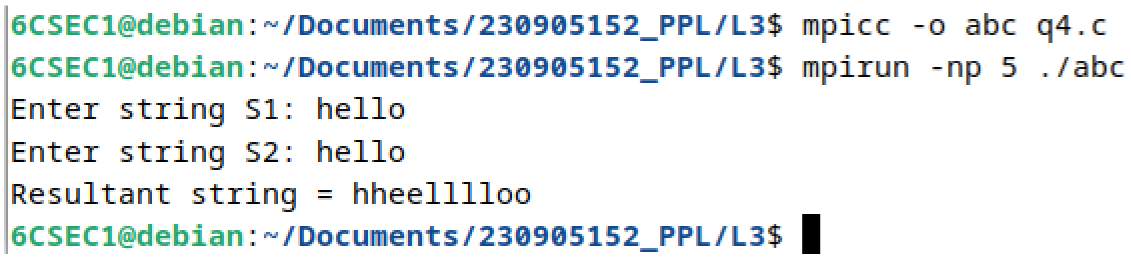
Q4.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mpi.h>

int main(int argc, char *argv[]) {
    int rank, size;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
```

```c
char *S1 = NULL, *S2 = NULL;
int len, chunk;
char *sub1, *sub2, *subres;
char *result = NULL;

if (rank == 0) {
    char buf1[1024], buf2[1024];
    printf("Enter string S1: ");
    fflush(stdout);
    scanf("%s", buf1);
    printf("Enter string S2: ");
    fflush(stdout);
    scanf("%s", buf2);

    len = strlen(buf1);
    if (len != strlen(buf2)) {
        printf("Strings must be same length\n");
        MPI_Abort(MPI_COMM_WORLD, 1);
    }
    if (len % size != 0) {
        printf("String length must be divisible by number of processes\n");
        MPI_Abort(MPI_COMM_WORLD, 1);
    }
    S1 = buf1;
    S2 = buf2;
}

MPI_Bcast(&len, 1, MPI_INT, 0, MPI_COMM_WORLD);
chunk = len / size;

sub1 = malloc(chunk);
sub2 = malloc(chunk);
subres = malloc(2 * chunk);

MPI_Scatter(S1, chunk, MPI_CHAR, sub1, chunk, MPI_CHAR, 0, MPI_COMM_WORLD);
MPI_Scatter(S2, chunk, MPI_CHAR, sub2, chunk, MPI_CHAR, 0, MPI_COMM_WORLD);

for (int i = 0; i < chunk; i++) {
    subres[2 * i] = sub1[i];
    subres[2 * i + 1] = sub2[i];
}
```

```
    if (rank == 0) result = malloc(2 * len + 1);
    MPI_Gather(subres, 2 * chunk, MPI_CHAR, result, 2 * chunk, MPI_CHAR, 0,
MPI_COMM_WORLD);

    if (rank == 0) {
        result[2 * len] = '\0';
        printf("Resultant string = %s\n", result);
        free(result);
    }

    free(sub1);
    free(sub2);
    free(subres);
    MPI_Finalize();
    return 0;
}
```

Output:



```
6CSEC1@debian:~/Documents/230905152_PPL/L3$ mpicc -o abc q4.c
6CSEC1@debian:~/Documents/230905152_PPL/L3$ mpirun -np 5 ./abc
Enter string S1: hello
Enter string S2: hello
Resultant string = hheelllloo
6CSEC1@debian:~/Documents/230905152_PPL/L3$
```

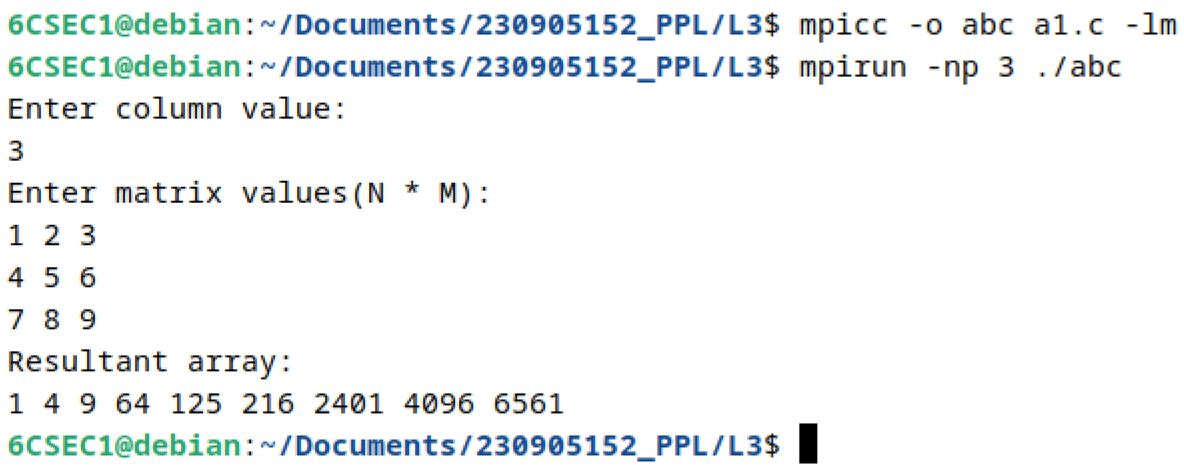
A1.

```
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>
#include <math.h>

int main(int argc, char *argv[]) {
    int rank, size;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    int M;
    int *array = NULL;
    int *sub = NULL;
    int *res = NULL;
    int *final = NULL;
```

```c
    if (rank == 0) {
        fprintf(stdout, "Enter column value: \n");
        fflush(stdout);
        scanf("%d", &M);
        array = malloc(size * M * sizeof(int));
        fprintf(stdout, "Enter matrix values(N * M): \n");
        fflush(stdout);
        for (int i = 0; i < size * M; i++) {
            scanf("%d", &array[i]);
        }
    }

    MPI_Bcast(&M, 1, MPI_INT, 0, MPI_COMM_WORLD);
    sub = malloc(M * sizeof(int));
    res = malloc(M * sizeof(int));

    MPI_Scatter(array, M, MPI_INT, sub, M, MPI_INT, 0, MPI_COMM_WORLD);

    int power = rank + 2;
    for (int i = 0; i < M; i++) {
        res[i] = (int)pow(sub[i], power);
    }

    if (rank == 0) {
        final = malloc(size * M * sizeof(int));
    }

    MPI_Gather(res, M, MPI_INT, final, M, MPI_INT, 0, MPI_COMM_WORLD);

    if (rank == 0) {
        printf("Resultant array:\n");
        for (int i = 0; i < size * M; i++) {
            printf("%d ", final[i]);
        }
        printf("\n");
        free(array);
        free(final);
    }

    free(sub);
    free(res);
    MPI_Finalize();
    return 0;
```

```
}
```

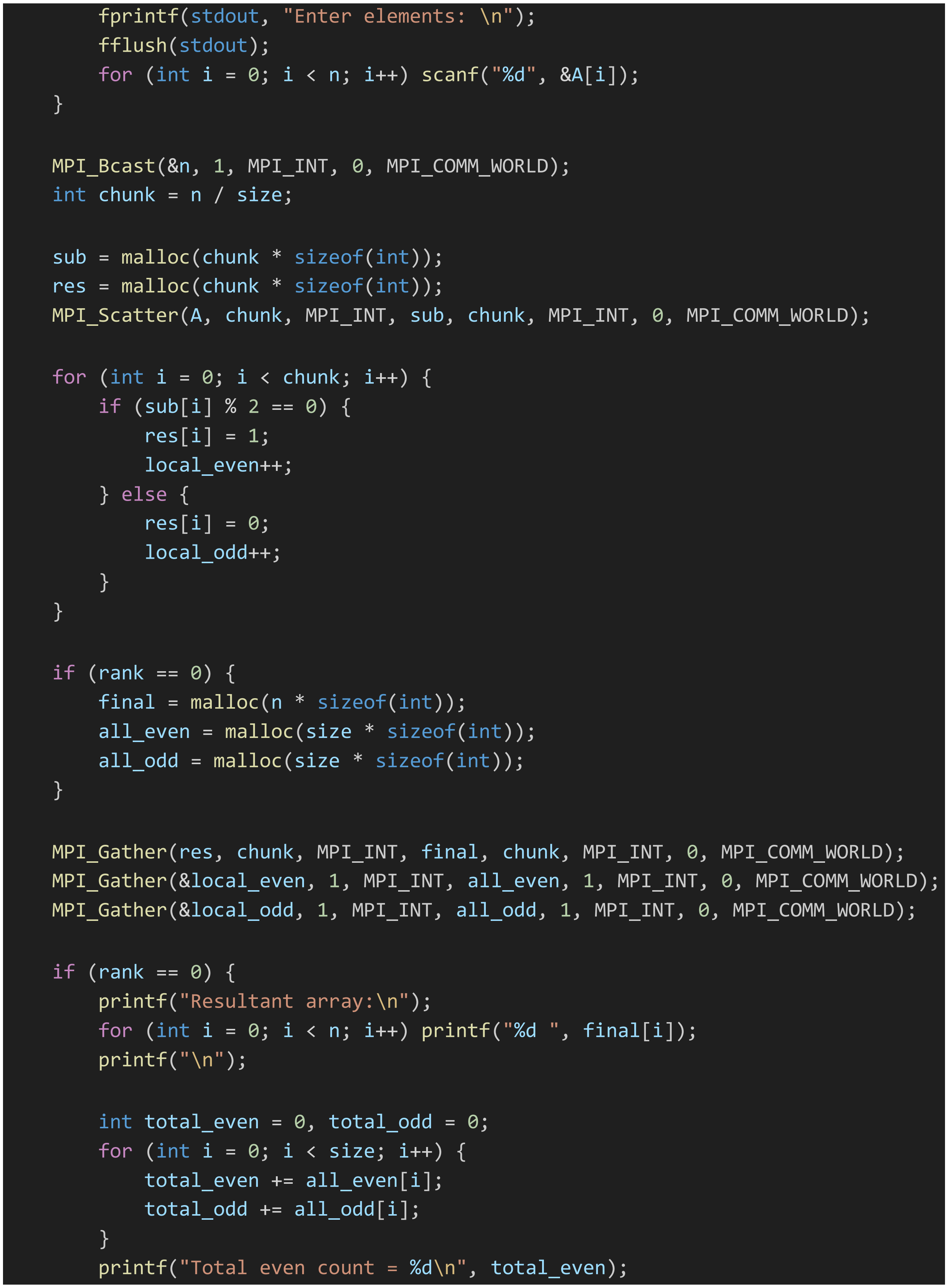
Output:



```
6CSEC1@debian:~/Documents/230905152_PPL/L3$ mpicc -o abc a1.c -lm
6CSEC1@debian:~/Documents/230905152_PPL/L3$ mpirun -np 3 ./abc
Enter column value:
3
Enter matrix values(N * M):
1 2 3
4 5 6
7 8 9
Resultant array:
1 4 9 64 125 216 2401 4096 6561
6CSEC1@debian:~/Documents/230905152_PPL/L3$
```

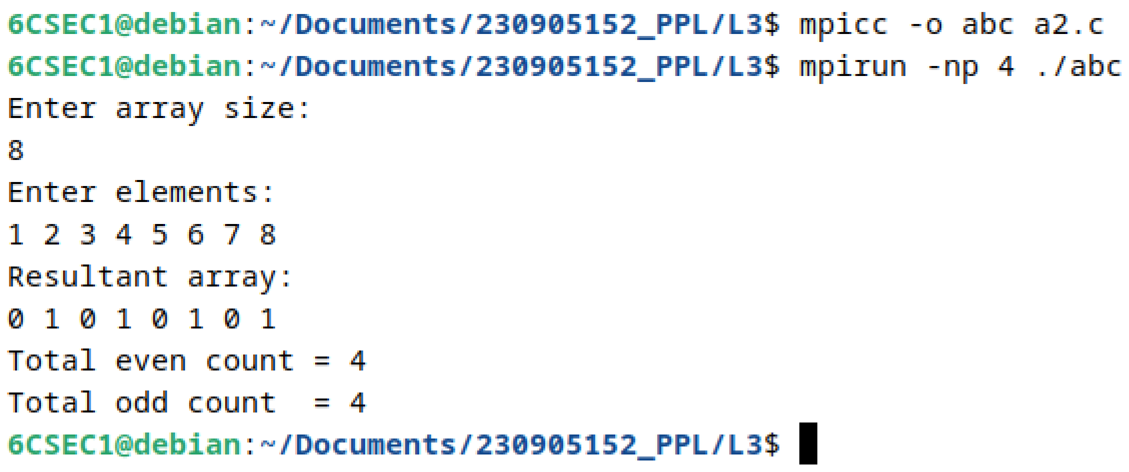
A2.

```c
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>

int main(int argc, char *argv[]) {
    int rank, size;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    int n;
    int *A = NULL;
    int *sub = NULL;
    int *res = NULL;
    int *final = NULL;
    int local_even = 0, local_odd = 0;
    int *all_even = NULL, *all_odd = NULL;

    if (rank == 0) {
        fprintf(stdout, "Enter array size: \n");
        fflush(stdout);
        scanf("%d", &n);
        if (n % size != 0) {
            printf("Array size must be divisible by number of processes\n");
            MPI_Abort(MPI_COMM_WORLD, 1);
        }
        A = malloc(n * sizeof(int));
```



```c
        fprintf(stdout, "Enter elements: \n");
        fflush(stdout);
        for (int i = 0; i < n; i++) scanf("%d", &A[i]);
    }

    MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
    int chunk = n / size;

    sub = malloc(chunk * sizeof(int));
    res = malloc(chunk * sizeof(int));
    MPI_Scatter(A, chunk, MPI_INT, sub, chunk, MPI_INT, 0, MPI_COMM_WORLD);

    for (int i = 0; i < chunk; i++) {
        if (sub[i] % 2 == 0) {
            res[i] = 1;
            local_even++;
        } else {
            res[i] = 0;
            local_odd++;
        }
    }

    if (rank == 0) {
        final = malloc(n * sizeof(int));
        all_even = malloc(size * sizeof(int));
        all_odd = malloc(size * sizeof(int));
    }

    MPI_Gather(res, chunk, MPI_INT, final, chunk, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Gather(&local_even, 1, MPI_INT, all_even, 1, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Gather(&local_odd, 1, MPI_INT, all_odd, 1, MPI_INT, 0, MPI_COMM_WORLD);

    if (rank == 0) {
        printf("Resultant array:\n");
        for (int i = 0; i < n; i++) printf("%d ", final[i]);
        printf("\n");

        int total_even = 0, total_odd = 0;
        for (int i = 0; i < size; i++) {
            total_even += all_even[i];
            total_odd += all_odd[i];
        }
        printf("Total even count = %d\n", total_even);
        printf("Total odd count = %d\n", total_odd);

        free(A);
        free(final);
        free(all_even);
        free(all_odd);
    }

    free(sub);
    free(res);
    MPI_Finalize();
    return 0;
}
```

Output:



```
6CSEC1@debian:~/Documents/230905152_PPL/L3$ mpicc -o abc a2.c
6CSEC1@debian:~/Documents/230905152_PPL/L3$ mpirun -np 4 ./abc
Enter array size:
8
Enter elements:
1 2 3 4 5 6 7 8
Resultant array:
0 1 0 1 0 1 0 1
Total even count = 4
Total odd count  = 4
6CSEC1@debian:~/Documents/230905152_PPL/L3$
```