

1.

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

```
/* We'll be using MPI routines, definitions, etc. */
```

```
#include "mpi.h"
```

```
int main(int argc, char** argv) {  
    int my_rank; /* My process rank */  
    int p; /* The number of processes */  
    float a; /* Left endpoint */  
    float b; /* Right endpoint */  
    int n; /* Number of trapezoids */  
    float h; /* Trapezoid base length */  
    float local_a; /* Left endpoint my process */  
    float local_b; /* Right endpoint my process */  
    int local_n; /* Number of trapezoids for */  
    /* my calculation */  
    float integral; /* Integral over my interval */  
    float total; /* Total integral */  
    int source; /* Process sending integral */  
    int dest = 0; /* All messages go to 0 */  
    int tag = 0;  
    MPI_Status status;
```

```
/* Let the system do what it needs to start up MPI */  
MPI_Init(&argc, &argv);
```

```
/* Get my process rank */  
MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
```

```
/* Find out how many processes are being used */  
MPI_Comm_size(MPI_COMM_WORLD, &p);
```

```
/* Add up the integrals calculated by each process */  
if (my_rank == 0) {  
    printf("Please input an a value: ");  
    scanf("%f", &a);  
    printf("Please input b value: ");  
    scanf("%f", &b);  
    printf("Please input n value: ");  
    scanf("%d", &n);
```

```
float Trap(float local_a, float local_b, int local_n,  
float h); /* Calculate local integral */
```

```
h = (b-a)/n; /* h is the same for all processes */
local_n = n/p; /* So is the number of trapezoids */
```

```
/* Length of each process' interval of
 * integration = local_n*h. So my interval
 * starts at: */
```

```
local_a = a + my_rank*local_n*h;
local_b = local_a + local_n*h;
```

```
integral = Trap(local_a, local_b, local_n, h);
total = integral;
for (source = 1; source < p; source++) {
    MPI_Recv(&integral, 1, MPI_FLOAT, source, tag,
    MPI_COMM_WORLD, &status);
    total = total + integral;
}
} else {
    MPI_Send(&integral, 1, MPI_FLOAT, dest,
    tag, MPI_COMM_WORLD);
}
```

```
/* Print the result */
if (my_rank == 0) {
    printf("With n = %d trapezoids, our estimate\n",
    n);
    printf("of the integral from %f to %f = %f\n",
    a, b, total);
}
```

```
/* Shut down MPI */
MPI_Finalize();
} /* main */
```

```
float Trap(
float local_a /* in */,
float local_b /* in */,
int local_n /* in */,
float h /* in */) {
```

```
float integral; /* Store result in integral */
float x;
int i;
```

```
float f(float x); /* function we're integrating */
```

```
integral = (f(local_a) + f(local_b))/2.0;
x = local_a;
for (i = 1; i <= local_n-1; i++) {
    x = x + h;
    integral = integral + f(x);
```

```

}
integral = integral*h;
return integral;
} /* Trap */

```

```

float f(float x) {
float return_val;
/* Calculate f(x). */
/* Store calculation in return_val. */
return_val = x*x;
return return_val;
} /* f */

```

The screenshot shows the Visual Studio Code editor with a C file named 'a.c'. The code implements a parallel trapezoidal integration using MPI. It includes a 'Trap' function that calculates the integral of a function f(x) = x^2 over the interval [a, b] using n trapezoids. The main function prompts the user for values of a, b, and n, and then calls the 'Trap' function to compute the integral. The terminal output shows the program being executed with MPI, and the user providing input values: a=1, b=4, and n=4. The program outputs the estimated integral value: 21.281250.

```

a.c - asignment1 - Visual Studio Code
EXPLORER
  OPEN EDITORS
    Welcome
  C a.c
  C b.c
  C c.c
  ASIGNMENT1
    .vscode
    a
    b
    c
    c.c
  C a.c
  C b.c
  C c
  C c.c

41 printf("Please input an a value: ");
42 scanf("%f", &a);
43 printf("Please input b value: ");
44 scanf("%f", &b);
45 printf("Please input n value: ");
46 scanf("%d", &n);
47
48 float Trap(float local_a, float local_b, int local_n,
49 float h); /* Calculate local integral */
50
51
52 h = (b-a)/n; /* h is the same for all processes */
53 local_n = n/p; /* So is the number of trapezoids */
54
55 /* Length of each process' interval of
56 ...

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: bash
dmitriy@dmitriy-Extensa-2510:~/code/parralel/assignment1$ mpiexec -np 1 ./a
Please input an a value: 1
Please input b value: 4
Please input n value: 4
With n = 4 trapezoids, our estimate
of the integral from 1.000000 to 4.000000 = 21.281250
dmitriy@dmitriy-Extensa-2510:~/code/parralel/assignment1$

```

2.

```

#include <stdio.h>

```

```

/* We'll be using MPI routines, definitions, etc. */
#include "mpi.h"

```

```

double dotProduct(double *a, double *b, int n) {
int i;
double prod = 0.0;
for (i = 0; i < n; i++) {
prod += a[i]*b[i];
}
return prod;
}

```

```

int main(int argc, char** argv) {
int my_rank;
int p;
double prod;
const int root=0;
int number = 3,a[3],b[3],i=0,j = 0,n = 3,loc_n;
int scalar;
//Number of scans
// a[0] = 1;
// a[1] = 2;
// a[2] = 3;
// b[0] = 4;
// b[1] = 5;
// b[2] = 6;
/* Let the system do what it needs to start up MPI */
MPI_Init(&argc, &argv);

/* Get my process rank */
MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

/* Find out how many processes are being used */
MPI_Comm_size(MPI_COMM_WORLD, &p);

if (my_rank == 0)
{
printf("Enter x , y, z of a vector \n");
while(i<number)
{
scanf("%d",&a[i]);
i++;
}
printf("Enter x , y, z of b vector\n");
while(j<number)
{
scanf("%d",&b[j]);
j++;
}
printf("Enter a scalar \n");
scanf("%d" ,&scalar);

// MPI_Scatter(&a,3,MPI_INT,&a,3,MPI_INT,my_rank,MPI_COMM_WORLD);
// MPI_Scatter(&b,3,MPI_INT,&b,3,MPI_INT,my_rank,MPI_COMM_WORLD);

}
MPI_Bcast(&a, 3, MPI_INT, root, MPI_COMM_WORLD);
MPI_Bcast(&b, 3, MPI_INT, root, MPI_COMM_WORLD);
// else{
loc_n = n/p;
double local_a[loc_n];
double local_b[loc_n];
for(i = 0; i < loc_n; i++) {

```

```

local_a[i] = scalar * a[i + my_rank * loc_n];
local_b[i] = scalar * b[i + my_rank * loc_n];
}
double local_prod;
local_prod = dotProduct(local_a, local_b, loc_n);
//printf("%f\n", local_prod);
MPI_Reduce(&local_prod, &prod, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
//MPI_Reduce_scatter(&local_prod, &prod, &root, MPI_DOUBLE, MPI_SUM,
MPI_COMM_WORLD);
// float global_sum;
// MPI_Allreduce(&local_prod, &prod, 1, MPI_FLOAT, MPI_SUM,
// MPI_COMM_WORLD);
// }
if (my_rank == 0) {
printf("dotProduct = %f\n", prod);
}
MPI_Finalize();
return 0;

} /* main */

```

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER:** Shows the project structure with files 'a.c', 'b.c', and 'c.c' under the 'ASIGMENT1' folder.
- EDITOR:** Displays the code for 'a.c', which includes MPI headers, defines MPI data types, and implements a parallel dot product function 'dotProduct'.
- TERMINAL:** Shows the command prompt output, including the compilation command 'mpicc b.c -o b' and the execution command 'mpiexec -np 3 ./b'. The output shows the input values for vectors a and b, and the resulting dot product value '64.000000'.

3.

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

```
int main(int argc, char** argv) {
```

```
    MPI_Init(&argc, &argv);
```

```
    int rank, size;
```

```
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
```

```
    MPI_Comm_size(MPI_COMM_WORLD, &size);
```

```
    int number = rank + 1;
```

```
    int summ = 0;
```

```
    int diff = size / 2;
```

```
    int total = 0;
```

```
    MPI_Allreduce(&number, &total, 1, MPI_INT,
```

```
    MPI_SUM, MPI_COMM_WORLD);
```

```
    if(rank == 0){
```

```
        printf("%d\n", total);
```

```
    }
```

```
    MPI_Finalize();
```

```
    return 0;
```

```
}
```

The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows a project named 'c.c - asignment1' with files 'a.c', 'b.c', and 'c.c'. The main editor window displays the C code from the previous blocks. The bottom panel shows the TERMINAL with the following commands and output:

```
dmitriy@dmitriy-Extensa-2510:~/code/parralel/asignment1$ mpicc c.c -o c
dmitriy@dmitriy-Extensa-2510:~/code/parralel/asignment1$ mpiexec -np 4 ./c
10
dmitriy@dmitriy-Extensa-2510:~/code/parralel/asignment1$
```

The status bar at the bottom indicates the file is 'c.c' at line 25, column 2, with 446 characters selected. The system is Linux.