Parallelization when calculations depend on the neighbours

In lecture 7, we learned about trapezoidal quadrature. In this calculation, you will need the left and right side of each trapezoid, which means that if we do the splitting of the job by alternation in the same way as we did in lab 4, it would mean that you would have to calculate the function twice for each point, which is inefficient.

So, instead of splitting the job by alternation, we need to figure out a *contiguous* section for each process to do, and make sure to take care of the leftovers, if the number of partitions do not divide exactly by the number of processes.

We will calculate the same thing as in part 3 of lab 5,

$$\int_{x=\pi/2}^{2\pi} \sin\left(\frac{x}{2}\right) + 1$$

but this time using trapezoidal quadrature. Here is the serial code:

```
//trapezoidal
#include<stdio.h>
#include<math.h>

double f(double x){
      return (sin (x/2) +1);
}

int main(){

      double a,b,w,sum=0,y[2];
      int i,n;

      printf("Enter the number of intervals : ");
      scanf("%d",&n);

      a=M_PI/2;
      b=2*M_PI;

      //step length (width of each trapezoid)
      w=(b-a)/n;
      // Initialize the heights
      y[0]= f(a);

      for(i=1; i<=n; i++)
      {     y[i%2] = f(a+i*w);
            sum += y[i%2] + y[(i+1)%2];
      }
      // Multiply with half the width.
      sum = sum * w/2;

      printf("Answer : %f \n",sum);

      return 0;
}
```

To convert it to MPI, you will now need to figure out the start and end points for each part that each process is to calculate, and change the loop to go from the start to the end that you calculated.

Note: this code uses the maths library, so you need to compile it with the `-lm` compiler directive.