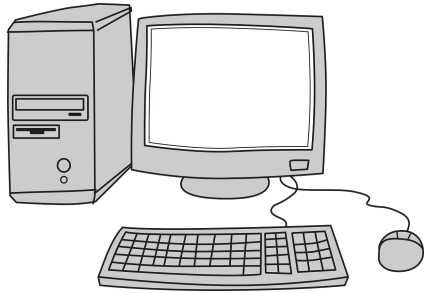
Final Exams

Exam 2

Exam 2

Given a C program that sorts an array of integers using Bubble Sort, your task is to parallelize this sorting algorithm using MPI. The provided code sequentially sorts the array, but your goal is to parallelize this process using MPI to take advantage of parallel computing. Develop a parallelized version where multiple processes independently sort different portions of the array. After sorting, each process should contribute to merging the sorted subarrays to obtain the final sorted array.

You can use any MPI function learned in the course.



Exam 2

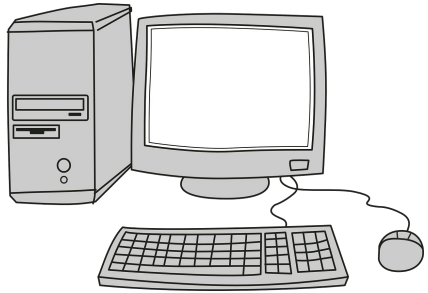
```
int main() {
    int array[ARRAY_SIZE];
    srand(12345); // Seed for reproducibility
    for (int i = 0; i < ARRAY_SIZE; i++)
        array[i] = rand() % 100;

    printf("Original array: ");
    for (int i = 0; i < ARRAY_SIZE; i++)
        printf("%d ", array[i]);
    printf("\n");

    bubbleSort(array, ARRAY_SIZE);

    printf("Sorted array: ");
    for (int i = 0; i < ARRAY_SIZE; i++)
        printf("%d ", array[i]);
    printf("\n");
    return 0;
}
```

This is the main function.

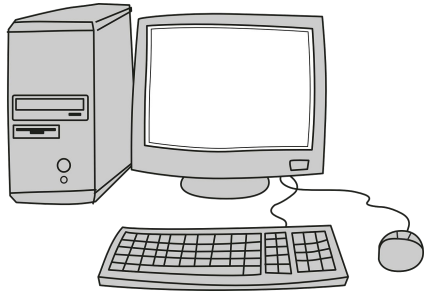


Exam 2

```
void bubbleSort(int arr[], int n) {
    for (int i = 0; i < n-1; i++) {
        for (int j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}
```

The bubble sort algorithm.

Consider `ARRAY_SIZE` will always be a multiple of the number of processes **P**.



Merge Technique (inner for loop)

- For each element in the subarray, **`temp`** stores the value of the current element being considered for insertion into the sorted portion of the array.
- The variable **`k`** is initialized to the index of the last element in the sorted portion of the array that precedes the current element.
- While **`k`** is greater than or equal to 0 (ensuring we're within the bounds of the array) and the value of the element at index **`k`** is greater than **`temp`**, we shift elements to the right to make space for the **`temp`** element.

Merge Technique (inner for loop)

- Once we find the correct position (**`k`** + 1) for the **`temp`** element in the sorted portion of the array, we insert **`temp`** at that position.