```
void generate_test_data(int test_array[], int size){
    for(int i=0;i<size;i++)
        test_array[i]=random(0,10001);
}

void generate_test_data_sorted(int test_array[], int size){
    for(int i=0;i<size;i++)
        test_array[i]=random(0,10001);
    int buf[size];
    merge_sort(test_array,buf,size);
}

void generate_test_data_sorted_butOne(int test_array[], int size){
    for(int i=0;i<size;i++)
        test_array[i]=random(0,10001);
    int buf[size];
    merge_sort(test_array,buf,size);
    // generate object which is not sorted (most likely not!)
    test_array[random(0,size)] = random(0,10001);
}

unsigned long test_insertion_sort(int test_array[], int size){
    long start = millis();
    insertion_sort(test_array,size);
    return millis()-start;
}
unsigned long test_merge_sort(int test_array[],int size){
    int buf[size];
    long start = millis();
    merge_sort(test_array,buf,size);
    return millis()-start;
}
void insertion_sort(int a[], int size){
    for(int i = 1; i < size; i++) {
        int j = i;
        while (j > 0 && a[j - 1] > a[j]) {
            // Swap a[j] and a[j - 1]
            int tmp = a[j];
            a[j] = a[j - 1];
            a[j - 1] = tmp;
            j--;
        }
    }
}

void merge_sort(int a[], int b[], int size)
{
    partial_merge_sort(a, b, 0, size);
}

void partial_merge_sort(int a[], int b[],int begin, int end){
    if(end - begin < 2)
        return;
    // Split and sort
    int mid = begin + (end - begin) / 2;
```

```
    partial_merge_sort(a, b, begin, mid);
    partial_merge_sort(a, b, mid, end);
    // Merge and copy
    merge(a, b, begin, mid, end);
    copy(a, b, begin, end);
}

void merge(int a[],int b[],int begin, int mid,int end){
    int i_begin = begin;
    int i_mid = mid;
    for(int j =begin;j<end;j++)
        if(i_begin < mid&& (i_mid >=end||a[i_begin] <=a[i_mid]))
            b[j] = a[i_begin++];
        else
            b[j] = a[i_mid++];
}
void copy(int a[], int b[], int begin, int end){
    for(int k = begin; k < end; k++)
        a[k] = b[k];
}
void setup(){
  // time to open serial monitor
  delay(5000);
    Serial.begin(9600);
    Serial.println("Laufzeitanalyse Unsorted");
    Serial.println("================");
    Serial.println(" ");

    for(int i = 100; i<1000;i+=100){
        int test_data_1[i];
        generate_test_data(test_data_1,i);
        unsigned long time_is = test_insertion_sort(test_data_1,i);
        int test_data_2 [i];
        generate_test_data(test_data_2,i);
        unsigned long time_ms = test_merge_sort(test_data_2,i);
        Serial.print("Anzahl Elemente: ");
        Serial.println(i);
        Serial.println("  Geschwindigkeit:");
        Serial.print("    InsertionSort : ");
        Serial.print(time_is);
        Serial.println(" ms");
        Serial.print("    MergeSort : ");
        Serial.print(time_ms);
        Serial.println(" ms");
    }

    Serial.println(" ");
    Serial.println(" ");
    Serial.println("Laufzeitanalyse Sorted");
    Serial.println("================");
    Serial.println(" ");

    for(int i = 100; i<1000;i+=100){
        int test_data_1[i];
        generate_test_data_sorted(test_data_1,i);
```

```
        unsigned long time_is = test_insertion_sort(test_data_1,i);
        int test_data_2 [i];
        generate_test_data_sorted(test_data_2,i);
        unsigned long time_ms = test_merge_sort(test_data_2,i);
        Serial.print("Anzahl Elemente: ");
        Serial.println(i);
        Serial.println("  Geschwindigkeit:");
        Serial.print("    InsertionSort : ");
        Serial.print(time_is);
        Serial.println(" ms");
        Serial.print("    MergeSort : ");
        Serial.print(time_ms);
        Serial.println(" ms");
    }

    Serial.println(" ");
    Serial.println(" ");
    Serial.println("Laufzeitanalyse Sorted but one");
    Serial.println("===============");
    Serial.println(" ");

    for(int i = 100; i<1000;i+=100){
        int test_data_1[i];
        generate_test_data_sorted_butOne(test_data_1,i);
        unsigned long time_is = test_insertion_sort(test_data_1,i);
        int test_data_2 [i];
        generate_test_data_sorted_butOne(test_data_2,i);
        unsigned long time_ms = test_merge_sort(test_data_2,i);
        Serial.print("Anzahl Elemente: ");
        Serial.println(i);
        Serial.println("  Geschwindigkeit:");
        Serial.print("    InsertionSort : ");
        Serial.print(time_is);
        Serial.println(" ms");
        Serial.print("    MergeSort : ");
        Serial.print(time_ms);
        Serial.println(" ms");
    }
}

void loop(){

}
```