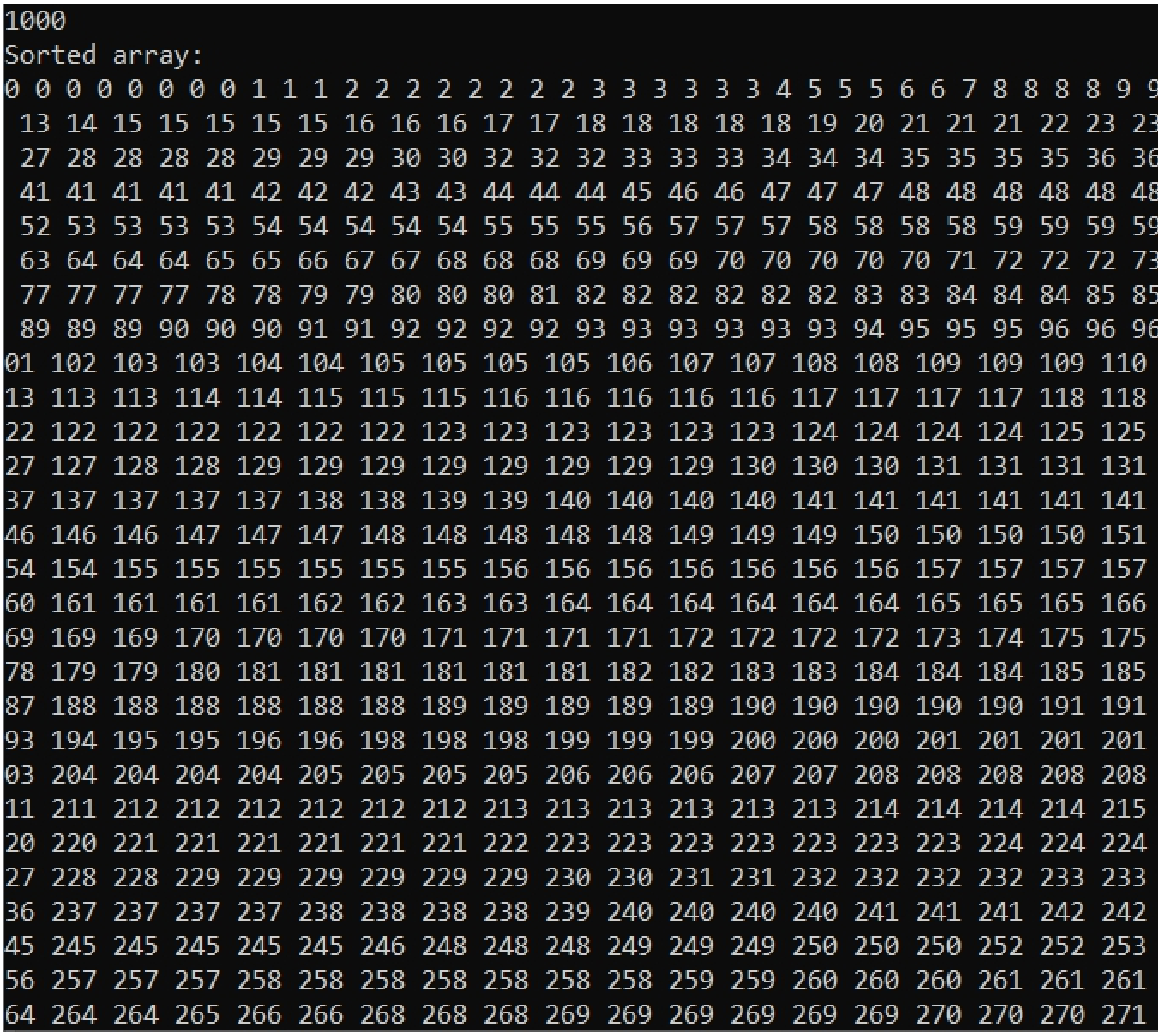
Sort a given set of N integer elements using Merge Sort technique and compute its time taken.
Run the program for different values of N and record the time taken to sort.

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

#include <time.h>
double time_spent = 0.0;
int n;


void merge(int arr[], int p, int q, int r) {


  int n1 = q - p + 1;
  int n2 = r - q;

  int L[n1], M[n2];

  for (int i = 0; i < n1; i++)
    L[i] = arr[p + i];
  for (int j = 0; j < n2; j++)
    M[j] = arr[q + 1 + j];


  int i, j, k;
  i = 0;
  j = 0;
  k = p;


  while (i < n1 && j < n2) {
    if (L[i] <= M[j]) {
      arr[k] = L[i];
      i++;
    } else {
      arr[k] = M[j];
      j++;
    }
    k++;
  }


  while (i < n1) {
    arr[k] = L[i];
    i++;
    k++;
  }

  while (j < n2) {
    arr[k] = M[j];
    j++;
    k++;
  }
}


void mergeSort(int arr[], int l, int r) {
  if (l < r) {


    int m = l + (r - l) / 2;

    mergeSort(arr, l, m);
    mergeSort(arr, m + 1, r);

    merge(arr, l, m, r);
  }
}


void printArray(int arr[], int size) {
  for (int i = 0; i < size; i++)
    printf("%d ", arr[i]);
  printf("\n");
}


int main() {
```

```
int arr[100000];
printf("enter the number of elements\n");
scanf("%d",&n);
for(int i=0;i<n;i++){
arr[i]=rand()%300;
}
clock_t start,end;

start=clock();
  mergeSort(arr, 0, n - 1);
end=clock();

  printf("Sorted array: \n");

  printArray(arr, n);
  time_spent += (double)(end -start) / CLOCKS_PER_SEC;

    printf("Time elpased is %f seconds\n", time_spent);

}
```

Program:



```
1000
Sorted array:
0 0 0 0 0 0 0 0 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 4 5 5 5 6 6 7 8 8 8 8 9 9
 13 14 15 15 15 15 15 16 16 16 17 17 18 18 18 18 18 19 20 21 21 21 22 23 23
 27 28 28 28 28 29 29 29 30 30 32 32 32 33 33 33 34 34 34 35 35 35 35 36 36
 41 41 41 41 41 42 42 42 43 43 44 44 44 45 46 46 47 47 47 48 48 48 48 48 48
 52 53 53 53 53 54 54 54 54 54 55 55 55 56 57 57 57 58 58 58 58 59 59 59 59
 63 64 64 64 65 65 66 67 67 68 68 68 69 69 69 70 70 70 70 70 71 72 72 72 73
 77 77 77 77 78 78 79 79 80 80 80 81 82 82 82 82 82 82 83 83 84 84 84 85 85
 89 89 89 90 90 90 91 91 92 92 92 92 93 93 93 93 93 93 94 95 95 95 96 96 96
01 102 103 103 104 104 105 105 105 105 106 107 107 108 108 109 109 109 110
13 113 113 114 114 115 115 115 116 116 116 116 116 117 117 117 117 118 118
22 122 122 122 122 122 122 123 123 123 123 123 123 124 124 124 124 125 125
27 127 128 128 129 129 129 129 129 129 129 129 130 130 130 131 131 131 131
37 137 137 137 137 138 138 139 139 140 140 140 140 141 141 141 141 141 141
46 146 146 147 147 147 148 148 148 148 148 149 149 149 150 150 150 150 151
54 154 155 155 155 155 155 155 156 156 156 156 156 156 156 157 157 157 157
60 161 161 161 161 162 162 163 163 164 164 164 164 164 164 165 165 165 166
69 169 169 170 170 170 170 171 171 171 171 172 172 172 172 173 174 175 175
78 179 179 180 181 181 181 181 181 181 182 182 183 183 184 184 184 185 185
87 188 188 188 188 188 188 189 189 189 189 189 190 190 190 190 190 191 191
93 194 195 195 196 196 198 198 198 199 199 199 200 200 200 201 201 201 201
03 204 204 204 204 205 205 205 205 206 206 206 207 207 208 208 208 208 208
11 211 212 212 212 212 212 212 213 213 213 213 213 213 214 214 214 214 215
20 220 221 221 221 221 221 221 222 223 223 223 223 223 223 223 224 224 224
27 228 228 229 229 229 229 229 229 230 230 231 231 232 232 232 232 233 233
36 237 237 237 237 238 238 238 238 239 240 240 240 240 241 241 241 242 242
45 245 245 245 245 245 246 248 248 248 249 249 249 250 250 250 252 252 253
56 257 257 257 258 258 258 258 258 258 258 259 259 260 260 260 261 261 261
64 264 264 265 266 266 268 268 268 269 269 269 269 269 269 270 270 270 271
```