```cpp
//Sorting Algorithms - Comparison

#include<iostream>
#include<fstream>
#include<time.h>
#include<stdlib.h>
#include<iomanip>

using namespace std;

void swap(int &a, int &b)
{
    int t = a;
    a = b;
    b = t;
}

long int count=0;

//Insertion Sort: 1. Insertion Sort
void InsertionSort(int *a,int n)
{
    int i,j,key;
    for(j=1;j<=n-1;j++)
    {
        i = j-1;
        key = a[j];

        while(i>=0 && a[i]>key)
        {
            count++;
            a[i+1] = a[i];
            i = i-1;
        }
        count++;

        a[i+1] = key;
    }
}
```

```
40:
41: //Merge Sort: 1. Merge
42: void Merge(int a[], int p, int q, int r)
43: {
44:     int n1,n2;
45:
46:     n1 = q-p+1;
47:     n2 = r-q;
48:
49:     int *L, *R;
50:
51:     L = new int[n1+1];
52:     R = new int[n2+1];
53:
54:     for(int i=0,j=0,k=p;k<=r;k++)
55:     {
56:         count++;
57:         if(k<=q)
58:         {
59:             L[i] = a[k];
60:             i++;
61:         }
62:         else
63:         {
64:             R[j] = a[k];
65:             j++;
66:         }
67:     }
68:
69:     L[n1] = 99999;
70:     R[n2] = 99999;
71:
72:     for(int i=0,j=0,k=p;k<=r;k++)
73:     {
74:         count++;
75:         if(L[i]<R[j])
76:         {
77:             a[k] = L[i];
78:             i++;
```

```
 79:         }
 80:         else
 81:         {
 82:             a[k] = R[j];
 83:             j++;
 84:         }
 85:     }
 86:
 87:     delete(L);
 88:     delete(R);
 89: }
 90:
 91: //Merge Sort: 2. Merge Sort
 92: void MergeSort(int a[], int p, int r)
 93: {
 94:     if(p>=r)
 95:         return;
 96:     int m;
 97:     m = (p+r)/2;
 98:     count++;
 99:     MergeSort(a,p,m);
100:     MergeSort(a,m+1,r);
101:     Merge(a,p,m,r);
102: }
103:
104:
105: //Quick Sort: 1. Partition
106:
107: int Partition(int a[], int s, int e)
108: {
109:     int key,i,j,n,t;
110:
111:
112:     key=a[e];
113:     i=s-1;
114:
115:     for(j=s;j<=e-1;j++)
116:     {
117:         count++;
```

```
118:         if(a[j]<=key)
119:         {
120:             i++;
121:             swap(a[i],a[j]);
122:         }
123:     }
124:     count++;
125:     swap(a[i+1],a[j]);
126:     return i+1;
127: }
128: //Quick Sort: 2. Quick Sort
129: void QuickSort(int a[],int s,int  e)
130: {
131:
132:     if(s>=e)
133:         return;
134:     int m;
135:
136:     m=Partition(a,s,e);
137:     QuickSort(a,s,m-1);
138:     QuickSort(a,m+1,e);
139: }
140:
141: //Heap Sort: 1. MaxHeapify
142: void MaxHeapify(int *a, int n, int i)
143: {
144:     int left, right, max;
145:
146:     left = 2*i + 1;
147:     right = 2*i + 2;
148:
149:     if(left<=n-1 && a[left]>a[i])
150:     {
151:         max = left;
152:     }
153:     else
154:     {
155:         max = i;
156:     }
```

```
157:
158:     if(right<=n-1 && a[right]>a[max])
159:     {
160:         max = right;
161:     }
162:
163:     if(max!=i)
164:     {
165:         count++;
166:         swap(a[i],a[max]);
167:         MaxHeapify(a,n,max);
168:     }
169:     count++;
170: }
171:
172: //Heap Sort: 2. BuildMaxHeap
173: void BuildMaxHeap(int *a,int n)
174: {
175:     for(int i=n/2;i>=0;i--)
176:     {
177:         MaxHeapify(a,n,i);
178:     }
179: }
180:
181: //Heap Sort: 3. Heap Sort
182: void HeapSort(int *a,int n)
183: {
184:     BuildMaxHeap(a,n);
185:     for(int i=n-1;i>=1;i--)
186:     {
187:         count++;
188:         swap(a[0],a[i]);
189:         n--;
190:         MaxHeapify(a,n,0);
191:     }
192: }
193:
194:
195: int main()
```

```
196: {
197:
198:
199:     cout<<showpoint<<setprecision(12);
200:
201:     ofstream outf;
202:     ifstream inf;
203:
204:     int n;
205:     int *a;
206:
207:     cout<<"\nEnter n:";
208:     cin>>n;
209:
210:     a = new int [n];
211:
212:
213:     //Loading numbers to input file
214:     outf.open("in.txt");
215:     for(int i=0;i<n;i++)
216:     {
217:         outf<<"\t"<<rand()%n;
218:     }
219:     outf.close();
220:
221:     //Insertion Sort
222:
223:     //Reading input in array from input file
224:
225:     inf.open("in.txt");
226:     for(int i=0;i<n;i++)
227:     {
228:         inf>>a[i];
229:     }
230:     inf.close();
231:
232:
233:
234:     count = 0;
```

```
235:     InsertionSort(a,n);
236: 
237: 
238:     //Writing sorted numbers to output file
239:     outf.open("InsertOut.txt");
240:     for(int i=0;i<n;i++)
241:     {
242:         outf<<"\t"<<a[i];
243:     }
244:     outf.close();
245: 
246:     cout<<"\n\nInsertion Sort:";
247:     cout<<"\nTotal Active Operations: "<<count;
248: 
249:     //Merge Sort
250: 
251:     //Reading input in array from input file
252: 
253:     inf.open("in.txt");
254:     for(int i=0;i<n;i++)
255:     {
256:         inf>>a[i];
257:     }
258:     inf.close();
259: 
260:     count=0;
261:     MergeSort(a,0,n-1);
262: 
263:     //Writing sorted numbers to output file
264:     outf.open("MergeOut.txt");
265:     for(int i=0;i<n;i++)
266:     {
267:         outf<<"\t"<<a[i];
268:     }
269:     outf.close();
270: 
271:     cout<<"\n\nMerge Sort:";
272:     cout<<"\nTotal Active Operations: "<<count;
273: 
```

```
274:
275:      //Quick Sort
276:
277:      //Reading input in array from input file
278:
279:      inf.open("in.txt");
280:      for(int i=0;i<n;i++)
281:      {
282:          inf>>a[i];
283:      }
284:      inf.close();
285:
286:      count = 0;
287:      QuickSort(a,0,n-1);
288:
289:      //Writing sorted numbers to output file
290:      outf.open("QuickOut.txt");
291:      for(int i=0;i<n;i++)
292:      {
293:          outf<<"\t"<<a[i];
294:      }
295:      outf.close();
296:
297:      cout<<"\n\nQuick Sort:";
298:      cout<<"\nTotal Active Operations: "<<count;
299:
300:      //Heap Sort
301:
302:      //Reading input in array from input file
303:
304:      inf.open("in.txt");
305:      for(int i=0;i<n;i++)
306:      {
307:          inf>>a[i];
308:      }
309:      inf.close();
310:
311:      count = 0;
312:      HeapSort(a,n);
```

```
313:
314:     //Writing sorted numbers to output file
315:     outf.open("HeapOut.txt");
316:     for(int i=0;i<n;i++)
317:     {
318:         outf<<"\t"<<a[i];
319:     }
320:     outf.close();
321:
322:     cout<<"\n\nHeap Sort:";
323:     cout<<"\nTotal Active Operations: "<<count;
324:
325:     delete(a);
326: }
327:
328:
329:
```