```
 1: //Maximum Sub-Array - Comparison of brute-force method &
    divide & conquer methods
 2: #include<iostream>
 3: #include<fstream>
 4: #include<stdlib.h>
 5: #include<time.h>
 6: #include<iomanip>
 7:
 8: using namespace std;
 9:
10: long int count=0;
11:
12: struct SubArray
13: {
14:     int low, high;
15:     double sum;
16:
17:     SubArray()
18:     {}
19:
20:     SubArray(int l, int h, double s)
21:     {
22:         low = l;
23:         high = h;
24:         sum = s;
25:     }
26: };
27:
28: SubArray MaxCrossingSubArray(double *a,int low, int mid,
    int high)
29: {
30:     double leftSum, rightSum, Sum;
31:     int maxLeft, maxRight;
32:
33:     leftSum = -99999;
34:     Sum = 0;
35:
36:     for(int i=mid;i>=low;i--)
37:     {
```

```
38:         count++;
39:         Sum = Sum + a[i];
40:         if(Sum > leftSum)
41:         {
42:             leftSum = Sum;
43:             maxLeft = i;
44:         }
45:     }
46:
47:     rightSum = -99999;
48:     Sum = 0;
49:
50:     for(int i=mid+1;i<=high;i++)
51:     {
52:         count++;
53:         Sum = Sum + a[i];
54:         if(Sum > rightSum)
55:         {
56:             rightSum = Sum;
57:             maxRight = i;
58:         }
59:     }
60:
61:     return SubArray(maxLeft,maxRight,leftSum+rightSum);
62: }
63:
64: SubArray MaximumSubArrayDC(double *a, int low, int high)
65: {
66:     SubArray leftSubArray, rightSubArray, crossSubArray;
67:
68:
69:     if(low == high)
70:     {
71:         return SubArray(low,high,a[low]);
72:     }
73:
74:     int mid;
75:
76:     count++;
```

```
 77:     mid = (low+high)/2;
 78:     leftSubArray = MaximumSubArrayDC(a, low, mid);
 79:     rightSubArray = MaximumSubArrayDC(a, mid+1, high);
 80:     crossSubArray = MaxCrossingSubArray(a,low,mid,high);
 81:
 82:     //Finding max between the three
 83:     if(leftSubArray.sum>=rightSubArray.sum)
 84:     {
 85:         if(leftSubArray.sum>=crossSubArray.sum)
 86:         {
 87:             return leftSubArray;
 88:         }
 89:         else
 90:         {
 91:             return crossSubArray;
 92:         }
 93:     }
 94:     else
 95:     {
 96:         if(rightSubArray.sum>=crossSubArray.sum)
 97:         {
 98:             return rightSubArray;
 99:         }
100:         else
101:         {
102:             return crossSubArray;
103:         }
104:     }
105:
106:
107:
108: }
109:
110: SubArray MaximumSubArrayBF(double *a, int low, int high)
111: {
112:     double maxSum,Sum;
113:     int left,right;
114:
115:     maxSum = a[low];
```

```
116:     left = 0;
117:     right = 0;
118:
119:     for(int i=low;i<=high;i++)
120:     {
121:         Sum = 0;
122:         for(int j=i;j<=high;j++)
123:         {
124:             count++;
125:             Sum = Sum + a[j];
126:             if(Sum>maxSum)
127:             {
128:                 maxSum = Sum;
129:                 left = i;
130:                 right = j;
131:             }
132:         }
133:     }
134:
135:     return SubArray(left,right,maxSum);
136: }
137:
138: int main()
139: {
140:     cout<<showpoint<<setprecision(12);
141:
142:     int n;
143:     double *a;
144:
145:     cout<<"\nEnter n: ";
146:     cin>>n;
147:
148:     a = new double[n];
149:
150:     ofstream outf;
151:     ifstream inf;
152:
153:     srand((long int)clock());
154:
```

```
155:     //Loading numbers to input file
156:     outf.open("in.txt");
157:     for(int i=0;i<n;i++)
158:     {
159:         if(rand()%2==0)
160:             outf<<"\t"<<(rand()%(n*2))*-1;
161:         else
162:             outf<<"\t"<<rand()%(n*2);
163:     }
164:     outf.close();
165:
166:     //Reading input in array from input file
167:
168:     inf.open("in.txt");
169:     for(int i=0;i<n;i++)
170:     {
171:         inf>>a[i];
172:     }
173:     inf.close();
174:
175:
176:     SubArray max;
177:
178:     //Brute-Force Method
179:     count = 0;
180:     max = MaximumSubArrayBF(a,0,n-1);
181:
182:     cout<<"\n\nBrute-Force Method: ";
183:     cout<<"\nMaximum Sub Array: ("<<max.low<<",
    "<<max.high<<", "<<max.sum<<")";
184:     cout<<"\nTotal Active Operations: "<<count;
185:
186:     //Divide-and-Conquer Approach
187:     count = 0;
188:     max = MaximumSubArrayDC(a,0,n-1);
189:
190:     cout<<"\n\nDivide-and-Conquer Approach: ";
191:     cout<<"\nMaximum Sub Array: ("<<max.low<<",
    "<<max.high<<", "<<max.sum<<")";
```

```
192:     cout<<"\nTotal Active Operations: "<<count;
193:
194: }
195:
```