4.4 Core (50%)

1. Consider the behaviour of the cost function for the insertion sort algorithm (Cost(n) described in the analysis above) for n = 10, n = 100, n = 1000, n = 1 million, in each of the following cases:

Cost(n) =(1/2C+1/2D)n^2+ (3/2D1/2C)n2D

a) C = 4, D = 2
cost(10)=(½(4)+½(2))10^2+ (3/2(2)1/2(4))10*2(2)=540
cost(100)=(½(4)+½(2))100^2+ (3/2(2)1/2(4))100*2(2)=32,400
cost(1000)=(½(4)+½(2))1000^2+ (3/2(2)1/2(4))1000*2(2)=3,024,000
cost(1,000,000)=(½(4)+½(2))1,000,000^2+(3/2(2)1/2(4))1000*2(2)
=>3,000,024,000,000

b) C = 2, D = 20
cost(10)=(½(2)+½(20))10^2+ (3/2(20)1/2(2))10*2(20)=13,100
cost(100)=(½(2)+½(20))100^2+ (3/2(20)1/2(2))100*2(20)=230,000
cost(1000)=(½(2)+½(20))1000^2+ (3/2(20)1/2(2))1000*2(20)=312,200,000
cost(1,000,000)=(½(2)+½(20))1,000,000^2+(3/2(20)1/2(2))1000*2(20)
=>11,001,200,000,000

c) C=200, D=2
cost(10)=(½(2)+½(20))10^2+ (3/2(20)1/2(2))10*2(20)=22,100
cost(100)=(½(2)+½(20))100^2+ (3/2(20)1/2(2))100*2(20)=1,130,000
cost(1000)=(½(2)+½(20))1000^2+ (3/2(20)1/2(2))1000*2(20)=102,200,000
cost(1,000,000)=(½(2)+½(20))1,000,000^2+(3/2(20)1/2(2))1000*2(20)
=>101,001,200,000,000

2. Run each sorting algorithm on each of the data sets supplied and tabulate the results.

	Data 1	Data 2	Data 3	Data 4	Average
Sort 1	2998	18312	49203	272970	85870.75
Sort 2	224	4407	16637	106425	31923.25
Sort 3	447	1421	8411	131567	35461.5
Sort 4	2389	2563	8970	96123	27511.25
Sort 5	138	5717	121000	16198682	1556384.25
Sort 6	324	6502	232939	50005032	12561199.25
Sort 7	590	20768	952386	308109681	77270856.25

3. Sort the algorithms in the order of their speed.

Data 1

Slowest						Fastest
Sort 1	Sort 4	Sort 7	Sort 3	Sort 6	Sort 2	Sort 5

Data 2

Slowest						Fastest
Sort 7	Sort 1	Sort 6	Sort 5	Sort 2	Sort 4	Sort 3

Data 3

Slowest						Fastest
Sort 7	Sort 6	Sort 5	Sort 1	Sort 2	Sort 4	Sort 3

Data 4

Slowest						Fastest
Sort 7	Sort 6	Sort 5	Sort 1	Sort 3	Sort 2	Sort 4

Average

Slowest						Fastest
Sort 7	Sort 6	Sort 5	Sort 1	Sort 3	Sort 2	Sort 4

4. For sort numbers 5, 6, 7 (only): analyse each algorithm by hand using the sample analysing described above (i.e. find the innermost loop with the “basic operation” and write out and derive a simple cost function for each algorithm). Notice that sort 5 is just the insertion sort from earlier.

   Algorithm 5
   Cost(n)=C*(1/2n(n-1)) + D*((n-1)+(1/2n(n-1))+(n-1))
   Cost(n) =(1/2C+1/2D)n^2 + (3/2D1/2C)n-2D

   Algorithm 6
   Cost (n)=C*(1/2n(n-1)) + D*((n-1)+(n-1)+(n-1))

   Algorithm 7
   Cost (n)=C*((n-1))+D*((n-1)+(n-1)+(n-1))

4.5 Completion (30%)

1. Based on your analysis in Core Question 4 above, order sorting algorithms 5,6,7 according to their efficiency (computed by hand) and compare with your results when ordering them based according to their run times.

2. Do the two lists match or not? Why or why not. Please write in your report a discussion of why they match or why some of them do not match.

3. For each function f(n) and time t in the following table, determine the largest size n of a problem that can be solved in time t, assuming that the algorithm to solve the problem takes f(n) microseconds:

f(n)	1 second	1 minute	1 hour	1 day	1 month	1 year	1 century
$\log_2 n$	n=2^1,000,000	n=2^60,000,000	n=2^3,600,000,000	n=2^86,400,000,000	n=2^2,592,000,000,000	n=2^31,104,000,000,000	n=2^311,040,000,000,000
n	1,000,000	60,000,000	3,600,000,000	86,400,000,000	592,000,000,000	31,104,000,000,000	311,040,000,000,000
$n.\log_2 n$							

n^2	1,000	7,745.97	60,000	293,938.77	769,415.36	5,577,096.02	17,636,326.15
2^n	19.93	25.84	31.75	36.33	39.11	44.82	48.14

4.6 Challenge (20%)

1. The following lists some possible algorithm runtimes, in microseconds. Please use the definition of Order Notation above to sort these in order of their asymptotic efficiency (the fastest should be first). Justify your ordering by referring to the definition of O(·).

    a) $c_1(n) = 2^n$
    b) $c_2(n) = n.\log_2(n^{99})$
    c) $c_3(n) = n!$
    d) $c_4(n) = (n + 99)^3$
    e) $c_5(n) = (5n + 7)^7$

2. For each case find a list-size (i.e. a value of n) which will take longer than one minute to process.

    f) $c_1(n) = 2^n$
    g) $c_2(n) = n.\log_2(n^{99})$
    h) $c_3(n) = n!$
    i) $c_4(n) = (n + 99)^3$
    j) $c_5(n) = (5n + 7)^7$