Question1

a) $0<=100n^3+ 8n^2 + 4n<=cn^4$ for all $n>n_0$

Choose c = 1 and $n_0 = 6$

$100n^3 + 8n^2 + 4n = n^4$ for all n>= 6

b) $T(n) = 8T(n/2) + n^3$

$T(n/2) = 8T(n/4) + (n/2)^3$

$T(n/4) = 8T(n/8) + (n/4)^3$

$T(n/8) = 8T(n/16) + (n/8)^3$

$T(n) = 8T(n/2) + n^3$

$T(n) = 64T(n/4) + n^3 + 8(n/2)^3$

$T(n) = 512T(n/8) + n^3 + 8(n/2)^3+8(n/4)^3$

$T(n) = 4096T(n/16) + n^3 + 8(n/2)^3+8(n/4)^3+8(n/8)^3$

$T(n) = 2^3T(n/2^1) + n^3$

$T(n) = 2^6T(n/2^2) + n^3 + 8(n/2^1)^3$

$T(n) = 2^9T(n/2^3) + n^3 + 8(n/2^1)^3+8(n/2^2)^3$

$T(n) = 2^{12}T(n/2^4) + n^3 + 8(n/2^1)^3+8(n/2^2)^3+8(n/2^3)^3$

$T(n) = 2^{3k}T(n/2^k) + n^3 + 8\sum(n/2^i)^3$ from 1 to k -1

$T(n) = 2^{3k}T(n/2^k) + n^3 + 8n^3(1-(1/8)^k)/7$

$n = 2^k$ k = logn

notice

n^3x2^-3k

move k

kn^3 rest part is unimportant

N^3logn

c)

for (i = n ; i > 0; i /= 2) { times logn

for (j = 1; j < n; j ++) {   times n

for (k = 1; k < n; k += 2) { times n

sum += (i + j * k) ; } } }

It is n

d)

[16, 6, 39, 21, 10, 21, 13, 7, 28, 19]

Selection sort:

[6,16, 39, 21, 10, 21, 13, 7, 28, 19]

[6,7, 39, 21, 10, 21, 13, 16, 28, 19]

[6,7, 10, 21, 39, 21, 13, 16, 28, 19]

[6,7, 10, 13, 39, 21, 21, 16, 28, 19]

[6,7, 10, 13, 16, 21, 21, 39, 28, 19]

[6,7, 10, 13, 16, 19, 21, 39, 28, 21]

[6,7, 10, 13, 16, 19, 21, 21, 28, 39]

Inserion Sort

[6,16, 39, 21, 10, 21, 13, 7, 28, 19]

[6,16, 21, 39, 10, 21, 13, 7, 28, 19]

[6,10, 16, 21, 39, 21, 13, 7, 28, 19]

[6,10, 16, 21, 21, 39, 13, 7, 28, 19]

[6,10, 13, 16, 21, 21, 39, 7, 28, 19]

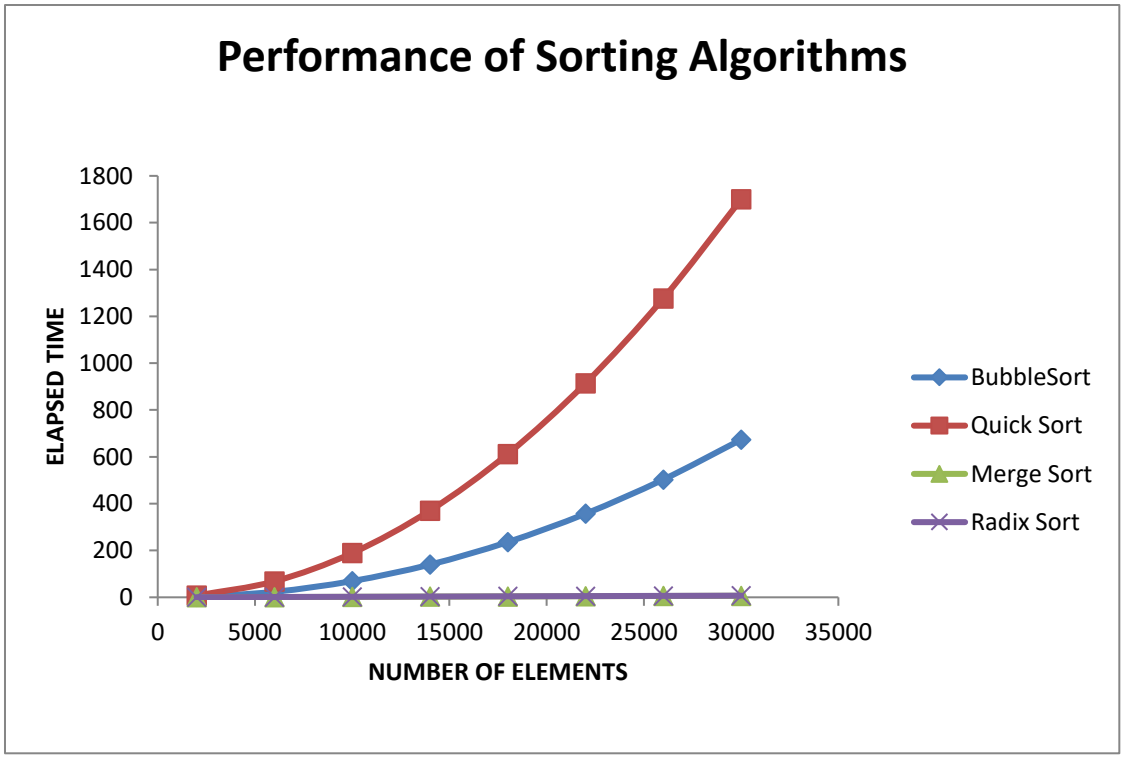
[6,7, 10, 13, 16, 21, 21, 39, 28, 19]

[6,7, 10, 13, 16, 21, 21, 28,39, 19]

[6,7, 10, 13, 16, 19, 21, 21,28, 39]

```
login as: bora.kurucu
bora.kurucu@dijkstra.ug.bcc.bilkent.edu.tr's password:
Last login: Sat Mar  9 19:34:07 2019 from 139.179.134.210
-bash-4.2$ make
g++ -c main.cpp -std=c++11
g++ -c Sorting.cpp -std=c++11
g++ main.o Sorting.o -o output -std=c++11
-bash-4.2$ ./output

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
------------------------
 part c time analysis of radix sort

Array Size      Time Elapsed
2000            0.466512
6000            1.42409
10000           2.37244
14000           3.33174
18000           4.27095
22000           5.21159
26000           6.16597
30000           7.11679
------------------------
 part c time analysis of bubble sort

Array Size      Time Elapsed    CompCount       MoveCount

2000            2.2575          2016396         2003042
6000            23.4717         18050936        18287550
10000           69.4109         50067036        49903904
14000           139.788         98097046        96704746
18000           236.09          162151665       162454882
22000           357.115         242186211       242081028
26000           503.193         338219910       339419224
30000           673.469         450217041       448444850
------------------------
 part c time analysis of quick sort

Array Size      Time Elapsed    CompCount       MoveCount

2000            7.70478         1999000         3998
6000            68.266          17997000        11998
10000           189.099         49995000        19998
14000           370.564         97993000        27998
18000           612.365         161991000       35998
22000           914.503         241989000       43998
26000           1277.05         337987000       51998
30000           1700.04         449985000       59998
------------------------
 part c time analysis of merge sort

Array Size      Time Elapsed    CompCount       MoveCount

2000            0.36186         11761           43904
6000            1.21231         41380           151616
10000           2.08351         73383           267232
14000           2.9707          107305          387232
```

```
18000           3.8887          141239          510464
22000           4.83729         174599          638464
26000           5.75222         211087          766464
30000           6.7037          248025          894464
-bash-4.2$
```



Bubble Sort is similar with the expected results. On average case it should be n^2, and it takes a lot time when compared with merge sort and quick sort. Also it's graph is similar with n^2,which makes sense.

Despite looking wrong, graph of quicksort is also makes sense. The values of integers in the array was less than 1000. Pivot choosing is bad. Pivot is the first one, so it is slow.Pivot couldn't fall in the middle, if fell nearly to the beginning all of the times. So it appeared in it's worst case O(n^2).

Radix sort with complexity 0(n) should have been smaller than mergeSort, what they were really close. That may be due to the opened program's, computer's speed and etc. However,both of them were pretty short as expected.

If the numbers were in ascending order, it wouldn't change anything for mergeSort. It would stil be in nlogn. For bubble sort it would change, the time complexity would be O(n^2),the number of moves and comparisons would be O(n^2).But here it is less than O(n^2). For quick sort, the pivot would fall to next one resulting in worst case O(n^2). It would be same O(n) for radix sort.