Answer 1

Entire working program-

```
#include "iostream"
#include <chrono>

using namespace std;
using namespace std::chrono;

void print_array(int *A,int size_of_array){
  for (int i=0;i<size_of_array;i++){
    cout<<A[i]<<" ";
  }
  cout<<endl;
}

void insertion_sort(int *A,int size_of_array){
  int i,key;
  for (int j=1;j<size_of_array;j++){
    i=j-1;
    key=A[j];
    while (key<A[i] & i>-1) {
      A[i+1]=A[i];
      i=i-1;
    }
    A[i+1]=key;
  }
}

void merge(int *A,int l,int mid, int r){
  /*
  A[l:mid] is sorted,
  A[mid+1:r] is sorted
  */
  int n1=mid-l+1;
  int n2=r-mid;
  int left[n1];
  int right[n2];

  for (int i=0;i<n1;i++){
    left[i]=A[l+i];
  }

  for (int j=0;j<n2;j++){
    right[j]=A[mid+j+1];
  }
  int i=0;
  int j=0;
  int k=l;
  while (i<n1 & j<n2){
    if (left[i]<right[j]){
      A[k]=left[i];
      i=i+1;
    }
    else{
      A[k]=right[j];
      j=j+1;
    }
    k=k+1;
  }

  while (i<n1){
    A[k]=left[i];
    k=k+1;
    i=i+1;
  }

  while (j<n2){
    A[k]=right[j];
    k=k+1;
    j=j+1;
  }
}

void merge_sort(int *A,int l,int r){
  /*
  Input: A[l...r].
  Initial call: l=0, r=len(A)-1.
  If len(A)==1, "if" condition at start would be false, and MERGE_SORT would return A directly
  */
  int mid;
  if (l<r){
    mid=(int)(l+r-1)/2;
    merge_sort(A,l,mid);
    merge_sort(A,mid+1,r);
    merge(A,l,mid,r);
  }
}

auto time_insertion_sort(int size_of_array){
  int A[size_of_array];
  for (int j=size_of_array;j>0;j--){
    A[size_of_array-j] = j;
  }
  //print_array(A,size_of_array);
  //cout<<"Insertion Sort ";
  auto start = high_resolution_clock::now();
  insertion_sort(A,size_of_array);
  auto stop = high_resolution_clock::now();
  auto duration = duration_cast<nanoseconds>(stop - start);
  //cout << "time "<<duration.count() << " ns"<<endl;
  //print_array(A,size_of_array);
  return duration;
}

auto time_merge_sort(int size_of_array){


  int B[size_of_array];
  for (int j=size_of_array;j>0;j--){
    B[size_of_array-j] = j;
  }
  //print_array(B,size_of_array);
  //cout<<"Merge Sort ";
  auto start = high_resolution_clock::now();
  merge_sort(B,0,size_of_array-1);
  auto stop = high_resolution_clock::now();
  auto duration = duration_cast<nanoseconds>(stop - start);
  //cout << "time "<<duration.count() << " ns"<<endl;
  //print_array(B,size_of_array);
  return duration;
}
int main() {
  cout<<"size_of_array"<<"\t"<<"insert sort time"<<"\t"<<"merge sort time"<<endl;

  for (int size_of_array=2;size_of_array<200000;size_of_array=size_of_array+1000){
    auto insert_time=time_insertion_sort(size_of_array);

    auto merge_time=time_merge_sort(size_of_array);
    cout<<size_of_array<<"\t"<<insert_time.count()<<"\t"<<merge_time.count()<<endl;
  }
  return 0;
}
```

Insertion sort code-

```
```

Input size n for which merge sort starts to beat insertion sort in terms of the worst-case running time-> n=31

Asymptotically, merge sort beats insertion sort-

Answer 3

$n+3 \in \Omega(n)$ **True**

$n+3 \in O(n^2)$ **True**

$n+3 \in \theta(n^2)$ **False**

$2^{n+1} \in O(n+1)$ **False**

$2^{n+1} \in \theta(2^n)$ **True**

Answer 4

$T(n) = 8*T(\frac{n}{2})+n = \theta(n^3)$

$T(n) = 8*T(\frac{n}{2})+n^2 = \theta(n^3)$

$T(n) = 8*T(\frac{n}{2})+n^3 = \theta(n^3*\log n)$

$T(n) = 8*T(\frac{n}{2})+n^4 = \theta(n^4)$

Answer 5

Recursion tree

- Total levels = $\log_2 n+1$
- At level j -> 8^j subproblems each of size $n/2^j$
- Computation at level j= $c*(8^j) * (n/2^j)$
  $= c* (n)* (4^j)$

- Total computation across all levels= $\sum_{j=0}^{\log_2 n} c* (n)* (4^j)$

  $= c* (n)*\sum_{j=0}^{\log_2 n} (4^j)$ (This is a geometric progression)

  $= c* (n)*4*(4^{(\log_2 n+1)}-1)/(4-1)$

  $= O(n*(4^{(\log_2 n+1)}-1))$
  $= O(n*(4*4^{(\log_2 n)})-n))$
  $= O(4*(n*n^{(\log_2 4)})-n))$
  $= O(4*(n^3)-n))$
  $= O(n^3)$

Substitution method

- Given: $T(n) = 8*T(\frac{n}{2})+n$
  Let the guess for $T(n)$ be $O(n^3)$.
- The substitution method requires us to prove that $T(n) \leq c*n^3$ for an appropriate choice of the constant $c > 0$
- We start by assuming that this bound holds for all positive m < n, in particular for m=n/2, yielding $T(\frac{n}{2}) \leq c*({\frac{n}{2}})^3$.
- Substituting into the recurrence yields
  $T(n) \leq 8*c*({\frac{n}{2}})^3+n$
  $T(n) \leq c*({n})^3+n$
  $T(n) = O({n})^3$

QED