Cognome e Nome _____

Matricola _____

Programmazione e Algoritmi I A.A. 2018/2019

Compito n° 1

Appello del 30 gennaio 2019

Esercizio 1 (8 punti)

Sia $T(n) = \begin{cases} 4T(n/2) + cn^2 & se\ n > 1 \\ d & se\ n = 1 \end{cases}$

con *c* e *d* costanti.

Si dia una stima esplicita (non ricorsiva) di T(n) facendo uso sia del **teorema generale** che del metodo di **sostituzione e induzione.**

Esercizio 2 (11 punti)

(3 punti) Si scriva un metodo di ricerca binaria (iterativo o ricorsivo) in pseudocodice o in Java

```
static int binarySearch (int[] a, int x)
```

che, preso come parametro un array **a** di numeri interi (ordinato in modo non decrescente) ed un intero **x**, restituisce la posizione di un elemento dell'array a uguale a x, se esiste, e **-1** altrimenti.

(8 punti) Due array a e b della stessa lunghezza n (ognuno dei quali non contiene elementi ripetuti) si dicono ***equiposizionali*** se per ogni posizione i=0,…,n-1, vale che a[i] è il j-esimo elemento più piccolo in a se e solo se b[i] è il j-esimo elemento più piccolo in b. Ad esempio, a={1, 8, 3} e b={10, 13, 11} sono equiposizionali.
Scrivere un metodo (in pseudocodice o in Java)

```
static boolean equiposizionali (int[] a, int[] b)
```

che, presi come parametro due array **a** e **b** di numeri interi, restituisce *true* se e solo se a e b sono della stessa lunghezza e sono ***equiposizionali***. Se entrambi valgono *null*, viene restituito *true*. Se solo uno dei due vale *null*, viene restituito *false*. Il metodo **non** deve modificare gli array.
Si analizzi la complessità temporale del metodo proposto: tale metodo può richiamare qualsiasi algoritmo visto a lezione e deve avere complessità temporale **O(n log n)** nel caso peggiore (*soluzioni con complessità temporale peggiore danno luogo a una valutazione minore, pari a un massimo 4 punti su 8*), dove n è la lunghezza degli array a e b in input.

Esercizio 3 (8 punti)

Si consideri il seguente array di numeri interi:

{18, 4, 10, 1, 20, 13, 17, 135, 65, 14}

Mostrare **passo-passo l'esecuzione dell'heap-sort** per ordinare l'array in modo non decrescente (compresa la fase iniziale di build-max-heap).

Esercizio 4 (8 punti)

A lezione abbiamo visto il tipo di dato **lista singolarmente concatenata, ordinata in modo non decrescente**

```
class List {
      int key;
      List next;
}
```

Con i metodi

```
static List insert (List l, int key) {
      List nuovo = new List();
      nuovo.key=key;
      List i=l;
      List prev=null;
      while (i!=null && i.key<key) {
            prev=i;
            i=i.next;
      }
      if (prev==null) {//devo inserire
                  //in PRIMA posizione
            nuovo.next=l;
            return nuovo;
      } else {//devo inserire dopo prev
            prev.next=nuovo;
            nuovo.next=i;
            return l;
      }
}
```

```
static List delete (List l, int key) {
      List i=l;
      List prev=null;
      while (i!=null && i.key<key) {
            prev=i;
            i=i.next;
      }
      if (i==null || i.key>key)return l;
      if (prev==null) {//cancello il
                        // PRIMO elemento
            return i.next;
      } else {
            prev.next=i.next;
            return l;
      }
}
```

Si consideri ora il tipo di dato **lista doppiamente concatenata, ordinata in modo non decrescente**

```
class List2 {
      int key;
      List2 prev;
      List2 next;
}
```

- **(4 punti)** Scrivere un metodo
  `static List2 insert (List2 l, int key)`
  che, presi come parametri una Lista **l** ed un intero **key**, inserisce l'intero nella lista e restituisce la nuova lista così ottenuta (identificata dal suo primo elemento).
- **(4 punti)** Scrivere un metodo
  `static List2 delete (List2 l, int key)`
  che, che, presi come parametri una Lista **l** ed un intero **key**, elimina (se presente) la prima occorrenza della chiave key dalla lista l. Viene restituita la nuova lista così ottenuta (identificata dal suo primo elemento).

Regole per lo svolgimento della prova scritta:

- Per svolgere il compito si hanno a disposizione **100** minuti.
- Scrivere **subito** nome, cognome, matricola su OGNI FOGLIO (**compreso questo**).
- Le risposte al primo esercizio devono essere date direttamente nel riquadro di questo foglio.
- Durante la prova scritta **non** è possibile abbandonare l'aula.
- Non è ammesso **per nessun motivo** comunicare in qualsiasi modo con altre persone
- È possibile consultare appunti, libri e dispense.
- Qualsiasi strumento elettronico di calcolo o comunicazione (telefoni cellulari, calcolatrici, palmari, computer, etc…) deve essere **completamente disattivato** e **depositato in vista sulla cattedra**
- Mettere in vista sul banco un valido documento di identità.

Cognome e Nome _____

Matricola _____

Programmazione e Algoritmi I A.A. 2018/2019

Compito n° 2

Appello del 30 gennaio 2019

Esercizio 1 (8 punti)

Sia $T(n) = \begin{cases} 9T(n/3) + cn^2 & se\ n > 1 \\ d & se\ n = 1 \end{cases}$

con *c* e *d* costanti.

Si dia una stima esplicita (non ricorsiva) di T(n) facendo uso sia del **teorema generale** che del metodo di **sostituzione e induzione.**

Esercizio 2 (11 punti)

(3 punti) Si scriva un metodo di ricerca binaria (iterativo o ricorsivo) in pseudocodice o in Java

```
static int binarySearch (int[] a, int x)
```

che, preso come parametro un array **a** di numeri interi (ordinato in modo non decrescente) ed un intero **x**, restituisce la posizione di un elemento dell'array a uguale a x, se esiste, e **-1** altrimenti.

(8 punti) Due array a e b della stessa lunghezza n (ognuno dei quali non contiene elementi ripetuti) si dicono ***equiposizionali*** se per ogni posizione i=0,…,n-1, vale che a[i] è il j-esimo elemento più piccolo in a se e solo se b[i] è il j-esimo elemento più piccolo in b. Ad esempio, a={1, 8, 3} e b={10, 13, 11} sono equiposizionali.
Scrivere un metodo (in pseudocodice o in Java)

```
static boolean equiposizionali (int[] a, int[] b)
```

che, presi come parametro due array **a** e **b** di numeri interi, restituisce *true* se e solo se a e b sono della stessa lunghezza e sono ***equiposizionali***. Se entrambi valgono *null*, viene restituito *true*. Se solo uno dei due vale *null*, viene restituito *false*. Il metodo **non** deve modificare gli array.
Si analizzi la complessità temporale del metodo proposto: tale metodo può richiamare qualsiasi algoritmo visto a lezione e deve avere complessità temporale **O(n log n)** nel caso peggiore (*soluzioni con complessità temporale peggiore danno luogo a una valutazione minore, pari a un massimo 4 punti su 8*), dove n è la lunghezza degli array a e b in input.

Esercizio 3 (8 punti)

Si consideri il seguente array di numeri interi:

{46, 13, 3, 1, 21, 14, 145, 45, 63, 24}

Mostrare **passo-passo l'esecuzione dell'heap-sort** per ordinare l'array in modo non decrescente (compresa la fase iniziale di build-max-heap).

Esercizio 4 (8 punti)

A lezione abbiamo visto il tipo di dato **lista singolarmente concatenata, ordinata in modo non decrescente**

```
class List {
      int key;
      List next;
}
```

Con i metodi

```
static List insert (List l, int key) {
      List nuovo = new List();
      nuovo.key=key;
      List i=l;
      List prev=null;
      while (i!=null && i.key<key) {
            prev=i;
            i=i.next;
      }
      if (prev==null) {//devo inserire
                  //in PRIMA posizione
            nuovo.next=l;
            return nuovo;
      } else {//devo inserire dopo prev
            prev.next=nuovo;
            nuovo.next=i;
            return l;
      }
}
```

```
static List delete (List l, int key) {
      List i=l;
      List prev=null;
      while (i!=null && i.key<key) {
            prev=i;
            i=i.next;
      }
      if (i==null || i.key>key)return l;
      if (prev==null) {//cancello il
                        // PRIMO elemento
            return i.next;
      } else {
            prev.next=i.next;
            return l;
      }
}
```

Si consideri ora il tipo di dato **lista doppiamente concatenata, ordinata in modo non decrescente**

```
class List2 {
      int key;
      List2 prev;
      List2 next;
}
```

- **(4 punti)** Scrivere un metodo
  `static List2 insert (List2 l, int key)`
  che, presi come parametri una Lista **l** ed un intero **key**, inserisce l'intero nella lista e restituisce la nuova lista così ottenuta (identificata dal suo primo elemento).
- **(4 punti)** Scrivere un metodo
  `static List2 delete (List2 l, int key)`
  che, che, presi come parametri una Lista **l** ed un intero **key**, elimina (se presente) la prima occorrenza della chiave key dalla lista l. Viene restituita la nuova lista così ottenuta (identificata dal suo primo elemento).

Regole per lo svolgimento della prova scritta:

- Per svolgere il compito si hanno a disposizione **100** minuti.
- Scrivere **subito** nome, cognome, matricola su OGNI FOGLIO (**compreso questo**).
- Le risposte al primo esercizio devono essere date direttamente nel riquadro di questo foglio.
- Durante la prova scritta **non** è possibile abbandonare l'aula.
- Non è ammesso **per nessun motivo** comunicare in qualsiasi modo con altre persone
- È possibile consultare appunti, libri e dispense.
- Qualsiasi strumento elettronico di calcolo o comunicazione (telefoni cellulari, calcolatrici, palmari, computer, etc…) deve essere **completamente disattivato** e **depositato in vista sulla cattedra**
- Mettere in vista sul banco un valido documento di identità.