Laboratorio di Algoritmi e Strutture Dati	Docente: A. Murano
Appello del 22 Gennaio 2007	
Laurea in Informatica	Università degli Studi di Napoli “Federico II”

Nome e Cognome Numero di Matricola:

Esercizi (esercizi 1, 2, 3 e 4)

1	2	3	4.	Totale
/6	/12	/6	/6	/30

Non utilizzate altri fogli, utilizzate soltanto lo spazio sottostante. Fogli differenti non saranno presi in considerazione per la correzione.

Gli studenti che hanno consegnato il progetto devono svolgere i primi due esercizi. Gli studenti che non hanno frequentato il corso e non hanno consegnato il progetto devono svolgere tutti gli esercizi. Tutti gli altri studenti devono svolgere i primi due esercizi

1. Si consideri una lista di numeri interi **Lista** implementata come lista doppiamente puntata non circolare implementata utilizzando la seguente struttura

```
struct elemento {
        struct elemento *prev;
        int inf;
        struct elemento *next;}

struct elemento *Lista;
```

   a. Si implementi in linguaggio C la funzione **togli_dispari** che, presa in input la lista ***Lista,** ricorsivamente elimini tutte le occorrenze di numeri dispari senza usare strutture dati aggiuntive e senza cambiare l’ordine degli elementi.
   b. Descrivere la complessità asintotica della funzione implementata

2. Relativamente all'esercizio 1, si considerino inoltre due grafi **G** e **H** grafi orientati pesati di **n** vertici 0,1,…, n-1 rappresentati con liste di adiacenza utilizzando la seguente struttura:

```
typedef struct graph {
        int   nv;
        edge   **adj; } graph;

        graph *G, *H;
```

```
typedef struct edge  {
        int   key;
        int   peso;
        struct  edge   *next; } edge;
```

   a. Scrivere poi in linguaggio C una funzione che preso in input il grafo **G** e **H**, restituisce un grafo **T** della stessa struttura di **G** e **H** e con lo stesso numero di nodi, tale che per ogni arco dal nodo **i** al nodo **j**, si inserisce l'arco in T se è presente sia in G che in H, la cui somma è però è presente nella lista Lista. In tal caso, il valore si elimina dalla lista.
   b. Descrivere la complessità asintotica della funzione implementata

3. Si consideri una coda di priorità per la gestione della coda di stampa di una rete implementata con una struttura dati heap **H[MAX]**. Si supponga di avere memorizzata la dimensione dell'heap in **heapsize** e di disporre delle seguenti funzioni:
    a. **void Heapify(int H[MAX], int el);**
    b. **void BuildHeap(int H[MAX]);**
    c. **void HeapSort(int H[MAX]);**
    d. **int ricerca (int H[MAX], int el);** \\ restituisce l'indice del vettore in cui si trova l'elemento **el;** e **-1** se l'elemento non è presente nel vettore

    Si implementi la funzioni **void annulla_lavoro(int H[MAX], int el)**, che prende in input l'heap **H[MAX]** e il lavoro **el** da eliminare ed elimina in lavoro **el** dall'heap **H[MAX]**. Descrivere la complessità asintotica della funzione annulla_lavoro.

4. Dopo una breve trattazione sugli algoritmi per il calcolo di percorsi minimi su un grafo, descrivere le caratteristiche fondamentali della Algoritmo di Dijkstra e la sua complessità asintotica. Descrivere inoltre in dettaglio la funzione “rilassamento” dell’algoritmo di Dijkstra, mostrandone le istruzioni. Infine, si mostri un funzionamento dell’algoritmo di Dijkstra tramite un esempio.