UNIVERSITÀ DI PISA

Concorso pubblico per titoli ed esami per l'ammissione al dottorato di ricerca in Informatica (anno 2006 - bandito con D.R. n. 01.4686 del 31 luglio 2006)
Totale posti a concorso n. 16 - borse di studio disponibili n. 8

1	**Il candidato svolga due esercizi a scelta dei quattro seguenti:** *A1)* Si consideri il classico modello RAM con l'assunzione che ciascuna cella di memoria e ciascun registro possano contenere *O(log n)* bit, dove *n* è la dimensione del problema computazionale in esame. Si progettino: 1. un algoritmo che, preso in ingresso un vettore A di n interi distinti, inverte l'ordine di tali elementi, ovvero rialloca gli elementi A[n], A[n-1], ..., A[2], A[1] nell'array in tale ordine; 2. un algoritmo che, preso in ingresso un vettore A di n interi distinti e un indice j dove $1 \leq j \leq n$, sposta i primi j elementi in fondo ad A (ossia, dopo lo spostamente, gli elementi contenuti nell'array risultano essere A[j+1], ..., A[n], A[1], ..., A[j]); 3. un algoritmo che, preso in ingresso un vettore A di n interi distinti compresi nell'intervallo da 1 a n+1, ne calcola l'unico mancante; Nella progettazione si **minimizzi** il costo degli algoritmi, sia in termini del numero di passi di esecuzione che del numero di celle di memoria utilizzate. Si discuta la complessità in tempo e in spazio degli algoritmi nel caso pessimo. *A2)* Si consideri un semplice linguaggio imperativo contenente i seguenti comandi: `c ::= skip	X=a	c;c	if b then c else c	while b do c`  dove: 1. `X` appartiene all'insieme `Var` delle variabili, 2. `a` appartiene all'insieme `Aexp` delle espressioni aritmetiche, 3. `b` appartiene all'insieme `Bexp` delle espressioni booleane la cui **semantica operazionale** sia stata fornita definendo: 1. un insieme $\Sigma = \{ \sigma \mid \sigma : \texttt{Var} \rightarrow \mathbb{N} \}$ di stati, 2. una funzione $\mathcal{A} : \texttt{Aexp} \times \Sigma \rightarrow \mathbb{N}$ per la valutazione delle espressioni aritmetiche, 3. una funzione $\mathcal{B} : \texttt{Bexp} \times \Sigma \rightarrow \mathbb{N}$ per la valutazione delle espressioni booleane e 4. una relazione $\hookrightarrow \subseteq (\texttt{c} \times \Sigma) \times \Sigma$ per la valutazione dei comandi, dove $\langle c,\sigma \rangle \hookrightarrow \sigma'$ indica che l'esecuzione completa del comando `c` nello stato σ termina nello stato σ'.

Si consideri un'estensione del linguaggio ottenuta estendendo l'insieme Aexp delle possibili espressioni aritmetiche in modo tale che la valutazione di un'espressione aritmetica possa generare un errore. Supponendo che le funzioni di valutazione $\mathcal{A}$ e $\mathcal{B}$ siano opportunamente estese aggiungendo una denotazione $\perp$ al loro codominio, si specifichi in che modo può essere estesa la definizione della relazione $\hookrightarrow$ per modellare correttamente la semantica operazionale del linguaggio esteso.

A3) Si consideri la pila di protocolli TCP/IP ed un protocollo ad essa appartenente. Si indichino le principali caratteristiche del protocollo, i meccanismi che esso offre, come esso viene interfacciato con gli altri protocolli della pila e come i meccanismi offerti possano essere utilizzati da altri programmi.

A4) Si consideri un sistema per l'allocazione dinamica della memoria che utilizza una *freelist*, in cui un allocatore ha a disposizione un'area di memoria e mantiene un puntatore alla lista dei blocchi liberi. Ogni elemento di tale lista è un blocco di memoria libera e contiene la dimensione del blocco (nei primi 4 byte) e il puntatore all'elemento successivo (nei successivi 4 byte).

Quando un utente richiede l'allocazione di un blocco di memoria, l'allocatore ispeziona la lista dei blocchi liberi e restituisce un puntatore a un blocco di dimensione sufficiente. All'utente viene restituito un puntatore all'area libera che segue immediatamente l'informazione sulla lunghezza del blocco.

Quando l'utente libera un blocco allocato, l'allocatore recupera la lunghezza salvata a suo tempo dai byte che precedono il puntatore dato e inserisce il blocco liberato nella lista dei blocchi liberi, per un suo eventuale successivo riuso.

Utilizzando le seguenti dichiarazioni C:

```
#define BYTE unsigned char    // singolo byte sulla piattaforma
#define MEMSIZE 131072        // dimensione della memoria
#define size_t unsigned int   // tipo delle dimensioni

struct freenode {             // dati di un blocco della freelist
  size_t size;                   // dimensione del blocco
  struct freenode *next;         // puntatore al prossimo blocco
};

struct allocnode {            // dati di un blocco allocato
  size_t size;                   // dimensione del blocco
};

unsigned char mem[MEMSIZE];   // area di memoria da gestire

struct freenode *freelist=(struct freenode *)mem;  // head della freelist

void init()                   // Inizializza la freelist
{
  freelist->size = MEMSIZE;      // inizialmente, tutta la memoria e' un
  freelist->next = NULL;         // singolo blocco libero
}
```

si scriva il codice C della funzione di deallocazione

```
void free(void *b)
```

che libera il blocco puntato da `b` e che unisce (quando possibile) blocchi di memoria libera.

2	**Il candidato descriva un settore della ricerca in informatica discutendone le principali applicazioni e/o problemi aperti.**