Fondamenti di Informatica – A.A. 2023-2024
Proff. Salvatore Andolina, Daniele Braga, Vincenzo Caglioti, Maristella Matera
Appello del 20/01/2024

POLITECNICO
MILANO 1863

Cognome:_____ Nome:_____ Matricola:_____ Voto: ___/30

Quesito	1	2.1	2.2	3	4	Tot
Punteggio Max	4	9	3	6	8	30
Valutazione						

Istruzioni:
- Il tempo massimo a disposizione per svolgere la prova è di 2h.
- È vietato comunicare, consultare appunti e utilizzare calcolatrici, telefoni, PC o qualsiasi dispositivo elettronico.
- Il voto minimo per superare la prova è 18.

Quesito 1 (4 punti). Codifica Binaria.

a) (1 punto) Di ciascuna delle seguenti operazioni di somma algebrica (*indicate con operandi rappresentati in base dieci per semplicità di lettura*), si dica se può essere correttamente eseguita rappresentando operandi e risultato in **complemento a 2 su 8 bit**. Si giustifichi brevemente la risposta (***N.B.:*** non serve eseguire le operazioni!).

1. –123 – 120
2. 123 – 120
3. –28 + 96
4. 122 + 102

b) (3 punti) Si mostri in dettaglio lo svolgimento delle operazioni 1. e 2. indicando esplicitamente se si verificano o non si verificano overflow e riporto perduto.

Quesito 2 (12 punti). File e strutture dati.

Ti è mai capitato di re-inserire qualche prodotto già presente nella tua "*lista della spesa*", magari digitale, solo perché usavi un editor che non rileva le ripetizioni? Con ***Spesa Facile*** sarà solo un brutto ricordo! Per esempio:

```
mele  2
pere  2
pane  1
mele  1
pesto 2
mele  4
olio  1
pere  1
```

nell'elenco soprastante, dove i prodotti sono rappresentati da un nome (*stringa di max 20 caratteri*) e una quantità da acquistare (*numero intero*), figurano in totale 7 mele e 3 pere, ma compaiono in linee duplicate, forse aggiunte distrattamente, in tempi diversi.

Punto 1 (9 punti). Si codifichi in C una funzione **`void`** `ripulisci(FILE * fin, FILE * fout)` che riceve dal chiamante due **file testuali** (già aperti) (`fin` in modalità lettura e contenente l'elenco dei prodotti potenzialmente duplicati, `fout` in modalità scrittura e inizialmente vuoto), **individua e unifica i prodotti duplicati,** e salva in `fout` l'elenco compattato.
Attenzione:

- Nel leggere il file in input la funzione si troverà verosimilmente a riversare i dati in una qualche struttura dati; si badi a progettarla in modo conveniente e a definirla in modo chiaro e comprensibile.
- Si può considerare definito un limite al numero di diversi prodotti inseribili(`#define MAX_PROD 100`) .
- Si ipotizzi che le linee del file di input siano sempre ben formate (con una stringa e un intero).
- Il file di output conterrà un nuovo elenco, senza linee duplicate, sostituite da un'unica linea con quantità uguale alla somma delle quantità di tutti i duplicati eliminati. Dall'esempio riportato sopra si otterrà:

```
mele 7
pere 3
pane 1
pesto 2
olio 1
```

Punto 2 (3 punti). Si codifichi un programma C (main) che riceve **da linea di comando** i nomi dei due file, invoca la funzione definita al punto precedente e stampa un messaggio per comunicare l'esito dell'operazione.
Si chiede di gestire esplicitamente gli eventuali errori nei parametri del main e nell'apertura del file.

Quesito 3 (6 punti). Funzioni Ricorsive.
Si codifichi *ricorsivamente* in C una funzione **`char`** `min_ch( char s[] )` che restituisce il *minimo* carattere di **`s`** (*nel consueto ordine alfabetico*). E' necessario, per il punteggio pieno, che il risultato sia individuato con il metodo ricorsivo. ***Esempi***: "MELO" → 'E' "BURRO" → 'B' "ZANGOLA" → 'A' "F" → 'F' "" → '\0'

Quesito 4 (8 punti). Liste dinamiche
Si consideri una lista dinamica di interi, definita come segue.

```
typedef struct nd { int valore;
                    struct nd * next; } Nodo;
typedef Nodo * Lista;
```

Si codifichi in C una funzione **`void`** `sdoppia( Lista lis, Lista * pPos, Lista * pNeg)` che riceve una lista `lis` e, modificando solo i collegamenti tra i nodi (senza allocare né deallocare memoria), costruisce e rende disponibili al chiamante due liste così definite: una (puntata da `pPos`) che contiene tutti gli elementi con valore positivo e una (puntata da `pNeg`) che contiene tutti gli elementi con valore negativo, preservandone l'ordine originale. L'invocazione di `sdoppia()` nel contesto di un programma chiamante può avvenire così:

```
Lista lis=NULL, pos=NULL, neg=NULL; /* Inizializzazioni: tre liste vuote */
…                                   /* "creazione" della lista lis, codice omesso */
sdoppia( lis, &pos, &neg );         /* Invocazione. pos e neg devono essere ancora vuote */
lis = NULL;                         /* Per evitare effetti collaterali - lis ora è "svuotata" */
```

Esempio: da `lis` → 1 → -2 → -3 → 4 → 5 → -6 ↵ costruisce `pos` → 1 → 4 → 5 ↵ e `neg` → -2 → -3 → -6 ↵