



| Cognome | Nome | Matricola | Voto: ... /30 |
|---------|------|-----------|---------------|
|---------|------|-----------|---------------|

| | | | | | | |
|----------|---|---|---|---|----|------|
| Quesito: | 1 | 2 | 3 | 4 | 5 | Tot. |
| Max: | 5 | 5 | 3 | 7 | 10 | 30 |
| Punti: | | | | | | |

Istruzioni:

- per contribuire alla valutazione finale, è necessario conseguire almeno 18/30;
- non è possibile consultare libri, appunti, la calcolatrice o qualsiasi dispositivo elettronico, né comunicare;
- tempo a disposizione: 2h 00m

Stile del codice C:

- non è necessario inserire direttive `#include`;
- i commenti non sono necessari, ma potrebbero essere utili in caso di errore;
- è consentito l'utilizzo di funzioni di libreria.

INIZIARE LA SOLUZIONE DI OGNI
ESERCIZIO SU UNA PAGINA

RESTITUIRE COMPILATO ANCHE
NEL CASO IN CUI CI SI RITIRA

Quesito 1 (5 punti)

Punteggio ottenuto .../5

Si consideri un sottoprogramma che, ricevuto in ingresso il puntatore a FILE associato ad un file di testo precedentemente aperto dal programma chiamante, crea una lista dinamica che riporta l'insieme dei valori interi contenuti nel file, e il numero di volte in cui ciascuno di essi compare, e restituisce la testa della lista. Il file di testo contiene una successione di numeri interi, separati l'uno dall'altro da caratteri '\n'.

Ad esempio, se il file contiene 5 0 -3 0 5 2 0 121 -18, la lista restituita sarà

$L \rightarrow (5,2) \rightarrow (0,3) \rightarrow (-3,1) \rightarrow (2,1) \rightarrow (121,1) \rightarrow (-18,1)$

Quesito 2 (5 punti)

Punteggio ottenuto .../5

Una linea spezzata è una lista dinamica di punti (coordinate intere). Una linea *spezzata aperta non degenera* è tale se i suoi punti sono almeno due, e sono tutti diversi tra loro. Data una linea spezzata aperta non degenera, la sua lunghezza è la somma delle distanze euclidee tra i punti consecutivi. Date due linee spezzate aperte non degeneri A e B, A è una *scorciatoia* di B se hanno stesso punto di inizio e di fine e A è più corta di B.

Partendo dal tipo di dato punto_t riportato, si sviluppi un sottoprogramma **Scorciatoia** che, ricevute in ingresso due spezzate aperte non degeneri, restituisca 1 se la prima è una scorciatoia della seconda, 0 altrimenti.

```
typedef struct _p {
    int x, y;
    struct _p * next;} punto_t;
```

Quesito 3 (3 punti)

Punteggio ottenuto .../3

Che cosa stampa a schermo il seguente programma?

```
#include <stdio.h>
void Funz(char *S);

int main()
{
    char S[100] = "abcdefghijklmnpq";
    Funz(S);
    return 0;
}

void Funz(char *S)
{
    if ('\0' != S[0])
    {
        Funz(S+1);
        printf("%c", *S);
    }
}
```

Quesito 4 (7 punti)

Punteggio ottenuto .../7

Si definisca un tipo “Boolean” e un tipo atto a rappresentare una Matrice di N x N interi, con N predefinita tramite una direttiva define.

Si scriva una funzione ricorsiva che, ricevendo in ingresso una Matrice e un eventuale altro parametro utile, restituisca True se la Matrice ricevuta in ingresso è simmetrica, False in caso contrario.

Quesito 5 (10 punti)

Punteggio ottenuto .../10

La notazione prefissa è un modo di rappresentare espressioni algebriche senza bisogno di parentesi.

Un’espressione prefissa può consistere (i) in un valore reale (rappresentato da un float) oppure

(ii) in una sequenza costituita da

- un operatore, rappresentato da un carattere ‘+’, ‘-’, ‘*’, oppure ‘/’
- un primo operando, costituito a sua volta da un’espressione prefissa
- un secondo operando, anch’esso costituito da un’espressione prefissa.

Ad esempio l’espressione * - 5.0 + 1.5 0.5 7.0 è il prodotto tra il primo operando (dato dalla differenza tra 5.0 e la somma tra 1.5 e 0.5) il cui valore pertanto è 3.0, e il secondo operando (il valore 7.0): perciò il valore dell’espressione è 21.0 .

Un’espressione prefissa è rappresentata tramite una lista dinamica di elementi:

```
typedef struct El {
    enum{oper, val} tipo;
    char operatore;
    float valore;
    struct El *next; } Elemento;
```

L’espressione dell’esempio è perciò rappresentata dalla seguente lista L:

$L \rightarrow (\text{oper}, '*', ...) \rightarrow (\text{oper}, '-', ...) \rightarrow (\text{val}, ..., 5.0) \rightarrow (\text{oper}, '+', ...) \rightarrow (\text{val}, ..., 1.5) \rightarrow (\text{val}, ..., 0.5) \rightarrow (\text{val}, ..., 7.0)$

Completare la funzione ricorsiva **Calcola** che riceve in ingresso un puntatore a un Elemento e restituisce il valore dell’espressione prefissa che comincia con l’elemento puntato. Ad esempio l’espressione che comincia con il quarto elemento della lista è la somma tra due operandi: 1.5 e 0.5. L’espressione che comincia con il terzo elemento vale 5.0. L’espressione che comincia con il secondo elemento è la differenza tra due operandi: il primo è l’espressione che comincia con il terzo elemento, e vale 5.0; il secondo è l’espressione che comincia con il quarto elemento (la somma tra 1.5 e 0.5, che vale 2.0). L’espressione che comincia con il secondo elemento perciò vale 3.0 .

float Calcola(Elemento *Punt)

{Elemento *PrimoOperando, *SecondoOperando;

Elemento *Cursore;

int;

if (Punt->tipo == val) **return** (.....);

PrimoOperando =;

.....

.....

.....

.....

SecondoOperando =;

if (Punt->operatore == ‘+’) **return** (..... +);

.....

.....

.....

}

/* caso base, non ricorsivo */

/* localizza il primo operando */

/* localizza il secondo operando */

