Fondamenti di Informatica - A.A. 2025-2026

Scuola di Ingegneria Industriale e dell'Informazione
Prof.ssa Cristiana Bolchini
Appello del **18/02/2026**

RESTITUIRE COMPILATO

POLITECNICO MILANO 1863

Cognome	**Nome**	**Cod. Persona**

Quesito:	1	2	3	4	5	6	Totale
Valutazione massima (in /30):	4	5	6	5	4	6	30
Valutazione quesito in decimi (/10):							

Istruzioni:

- gli esercizi devono essere risolti utilizzando il C **ANSI 89**, in linea con quanto fatto durante il corso;
- non è possibile consultare libri, appunti, la calcolatrice o qualsiasi dispositivo elettronico, tantomeno comunicare;
- **non è possibile scrivere a matita**;
- tempo a disposizione: 1h 50m

Stile del codice C:

- non è necessario inserire direttive `#include`;
- i commenti non sono necessari, ma potrebbero essere utili nel caso di errore;
- è possibile utilizzare sottoprogrammi di libreria.
- **iniziare ogni esercizio in una pagina nuova**

SCRIVERE SOLO ALL'INTERNO DELLE CORNICI

Queste informazioni non vengono in alcun modo considerate ai fini della valutazione
Sapevo già programmare:
◯ No, non è vero ◯ in C ◯ in C++/C# ◯ in Python ◯ in Java ◯ in PHP/Javascript ◯ in VB* ◯ in altro linguaggio
Ho frequentato (lezioni ed esercitazioni):
◯ occasionalmente ◯ fino alle prove in itinere (metà corso) ◯ fino alla fine ma con molte assenze ◯ sempre (occasionali assenze)

Sottoprogrammi per la gestione di liste concatenate semplici: qualora utili, si considerino già disponibili (e quindi non da sviluppare). Qua ci si riferisce a un campo intero, ma nel caso la lista debba gestire caratteri (o dati di natura diversa), ipotizzate di avere a disposizione il sottoprogramma corrispondente analogo. **Definite il tipo di dato `listtype` così come vi è utile, e mettete tale definizione all'inizio dell'esercizio.**

```
/* inserisce in testa alla lista elemento con valore specificato */
listtype * push(listtype *, int);
/* inserisce in coda alla lista elemento con valore specificato */
listtype * append(listtype *, int);
/* inserisce ordinatamente in lista elemento con valore specificato, in senso crescente */
listtype * increasing(listtype *, int);
/* inserisce ordinatamente in lista elemento con valore specificato, in senso decrescente */
listtype * decreasing(listtype *, int);
/* elimina dalla lista il primo elemento */
listtype * pop(listtype *);
/* elimina dalla lista tutti gli elementi con il valore indicato */
listtype * delete(listtype *, int);
/* restituisce il riferimento all'elemento che ha il valore indicato, se esiste, NULL altrimenti */
listtype * find(listtype *, int);
/* restituisce il numero di elementi nella lista */
int length(listtype *);
/* elimina la lista */
listtype * emptylist(listtype *);
```

Quesito 1 [4 pti]

Dati i due valori X = 1001101101101_{2MS} e Y = 11101101101_{2C2} effettuare la conversione in base 2, notazione complemento a 2 (2C2), di ognuno degli operandi sul numero **minimo** di bit necessari. Si effettuino quindi le operazioni X+Y e X-Y indicando esplicitamente se si verifica overflow o meno, e motivando la risposta. **Mostrare i passaggi fatti e motivare la risposta relativa all'overflow (senza passaggi/calcoli e motivazioni, l'esercizio non viene valutato).**

Riportare nello spazio sottostante la codifica di X_{2C2}, Y_{2C2} e i risultati finali delle operazioni (tutti i passaggi devono essere fatti sopra), utilizzando solo le caselle necessarie (**allineati a destra**) ed indicando se si è verificato overflow (segnare la casella corrispondente).

X_{2C2} | | | | | | | | | | | | | | | | | | | | | |

Y_{2C2} | | | | | | | | | | | | | | | | | | | | | |

$X+Y_{2C2}$ | | | | | | | | | | | | | | | | | | | | | | ☐ OVF

$X-Y_{2C2}$ | | | | | | | | | | | | | | | | | | | | | | ☐ OVF

Quesito 2 [5 pti]

Si considerino le seguenti varianti di un sottoprogramma `s2i` che riceve in ingresso una stringa e calcola e restituisce il valore intero corrispondente (simile a quanto fa il sottoprogramma di libreria standard `atoi`). Il contenuto della stringa, quando non vuota, è senz'altro una sequenza di cifre, non c'è mai un contenuto anomalo. È presente in tutti i casi una direttiva `#define BASE 10`.

Versione A

```
int s2i(char s[])
{
  int i, ris;

  ris = 0;
  i = 0;
  while(s[i] != '\0') {
    ris = ris +
      (s[i] - '0') * BASE;
    i++;
  }
  return ris;
}
```

Versione B

```
int s2i(char s[])
{
  int i, ris, pot;
  ris = 0;
  pot = 1;
  i = 0;
  while(s[i] != '\0') {
    ris = ris +
      (s[i] - '0') * pot;
    pot = pot * BASE;
    i++;
  }
  return ris;
}
```

Versione C

```
int s2i(char s[])
{
  int i, ris;

  ris = 0;
  i = 0;
  while(s[i] != '\0') {
    ris = ris * BASE +
      (s[i] - '0');
    i++;
  }
  return ris;
}
```

Versione D

```
int s2i(char s[])
{
  int i, ris;

  ris = 0;
  i = 0;
  do {
    ris = ris * BASE +
      (s[i] - '0');
    i++;
  } while(s[i] != '\0');

  return ris;
}
```

(4 pti) Tracciare l'esecuzione delle versioni con la stringa `s = "347"`, mostrando il valore delle variabili ad ogni iterazione.

Iter	i	s[i]	Versione A		Versione B		Versione C		Versione D	
			ris (prima)	ris (dopo)	ris (prima)	ris (dopo)	ris (prima)	ris (dopo)	ris (prima)	ris (dopo)
1	0	'3'								
2	1	'4'								
3	2	'7'								
return										

(1 pto) Quale/i versione/i è/sono corretta/e, spiegando le differenze tra la/e versione/i corretta/e e quella/e errata/e?

- ◯ Versione A
- ◯ Versione B
- ◯ Versione C
- ◯ Versione D

Quesito 3 [6 pti]

Scrivere un sottoprogramma che riceve come parametro una stringa che rappresenta il nome di un file contenente una sequenza di lunghezza indefinita (eventualmente vuota) di numeri interi positivi (è senz'altro così), separati da spazi. Il sottoprogramma conta quante volte un numero `n` presente nella sequenza compare esattamente `n` volte consecutivamente, e quante volte questo non accade e **trasmette** tali due risultati al chiamante. In caso di errore nell'accesso al file, il sottoprogramma trasmette i valori -1, -1. Lo stesso valore può comparire più volte nel file.

Esempio: Se il contenuto del file è `3 3 3 1 21 21 21 5 5 5 5 5 4 3 3`, il sottoprogramma `trasmette` al chiamante i conteggi 3 e 2, poiché i valori 3 (la prima volta), 1 e 5 compaiono il numero prescritto di volte consecutivamente (ossia, 3, 1 e 5) mentre 21 e 4 e 3 (la seconda volta) no (non compaiono 21 volte, 4 volte e 3 volte).

Quesito 4 [5 pti]

Le immagini digitali sono rappresentate come una griglia bidimensionale di pixel. Ogni pixel ha un colore, che viene codificato mediante la combinazione di tre canali: Rosso (**R**ed), Verde (**G**reen) e Blu (**B**lue). Ciascun canale ha un valore intero compreso nell'intervallo `[0, 255]`. Un pixel è quindi rappresentato da una tripletta di valori interi. Un modo alternativo per rappresentare tale colore del pixel è giustapporre i tre valori *rappresentati* in base esadecimale, ottenendo una stringa di 6 caratteri (ad esempio il colore rosso consiste della tripletta RGB `[255, 0, 0]` e la sua rappresentazione esadecimale corrisponde a `FF0000`). A questo proposito si consideri il tipo di dato per la rappresentazione di colori tramite i canali RGB, e la sequenza esadecimale, ed un ulteriore intero nel caso si voglia convertire il valore su una scala a monocromatica (bianco o nero):

```
#define N 6
#define INDEFINITO -1
#define BIANCO 255
#define NERO 0
typedef struct pixel_s {
   int r, g, b;      /* i tre canali Red Green e Blue*/
   char hex[N+1];    /* stringa esadecimale */
   int bw;           /* valore singolo quando bianco e nero*/
} pixel_t;
```

Scrivere un **sottoprogramma** `completaColore` che riceve in ingresso variabile di tipo `pixel_t` che contiene **o** solo le informazioni dei canali RGB (e.g., `r, g, b` con valori diversi da `-1` e con la stringa vuota), oppure solo la stringa esadecimale (e con i tre canali sono a valore `INDEFINITO`). Il sottoprogramma ha come scopo quello di completare la struttura dati con l'informazione mancante in modo tale che al termine ci sia la parte RGB **e** la stringa esadecimale (si ignori l'ulteriore campo per il bianco e nero). Il sottoprogramma *trasmette al chiamante* la variabile modificata. Si vedano gli esempi seguenti.

Input:	`255 0 0 ""`	`39 145 245 ""`	`-1 -1 -1 "389114"`	`-1 -1 -1 "D65C1E"`
Output:	`255 0 0 "FF0000"`	`39 145 245 "2791F5"`	`56 145 20 "389114"`	`214 92 30 "D65C1E"`

Quesito 5 [4 pti]

Quando si vuole convertire una immagine a colori in bianco e nero è possibile adottare un filtro che converte ogni pixel in un unico valore intero compreso nell'intervallo `[0, 255]`.

La conversione di ogni pixel dalle tre componenti RGB a un singolo valore intero nell'insieme `[0, 255]` viene effettuata utilizzando la trasformata di luminanza ITU-R 601-2 seguita dall'applicazione di soglia. Date le tre componenti, il valore di luminanza `l` è calcolato come:

$$l = r \cdot \frac{299}{1000} + g \cdot \frac{587}{1000} + b \cdot \frac{114}{1000}$$

I pixel con valore di luminanza **strettamente** maggiore di `127` vengono convertiti in bianco (`255`); gli altri pixel vengono convertiti in nero (`0`).

Scrivere un **sottoprogramma** `convertiBW` che, ricevuta in ingresso **un'immagine** rappresentata come un array bidimensionale di pixel, dove ogni pixel è rappresentato mediante la struttura dati del tipo sopra definito (`pixel_t`), e qualsiasi altro parametro ritenuto strettamente necessario, **modifica** l'immagine calcolando per ogni pixel il suo colore *bianco* (`255`) oppure *nero* (`0`) e assegnandolo all'opportuno campo `bw` presente. Sono presenti le seguenti direttive per la specifica delle dimensioni massime dell'immagine:

```
#define MAXROW ...
#define MAXCOL ...
```

Quesito 6 [6 pti]

Scrivere un sottoprogramma che riceve in ingresso una stringa `s`, di lunghezza **senz'altro pari** e contenente **esclusivamente lettere minuscole dell'alfabeto** (è senz'altro così). Il sottoprogramma **crea e restituisce** una nuova stringa creata nel modo di seguito descritto: la stringa `s` viene analizzata a partire dall'inizio considerando *coppie* di caratteri consecutivi s_i e s_{i+1}. Per ciascuna coppia, il sottoprogramma aggiunge alla nuova stringa tutte le lettere comprese tra s_i e s_{i+1}, estremi inclusi, secondo l'ordine alfabetico. Il risultato finale è la stringa ottenuta concatenando, per ogni coppia di caratteri consecutivi in `s`, l'intervallo di lettere corrispondente.

Esempi:

Input:	`"adhexxmp"`	`"aabbccde"`	`"esempi"`
Output:	`"abcdhgfexmnop"`	`"abcde"`	`"efghijklmnopqrsefghijklmponmlkji"`