	Politecnico di Milano Scuola di Ingegneria Industriale e dell'Informazione **FONDAMENTI DI INFORMATICA** Appello 4 Settembre 2018		COGNOME E NOME
	RIGA	COLONNA	CODICE PERSONA
			Spazio riservato ai docenti

- Il presente plico contiene 5 esercizi e **deve essere debitamente compilato con cognome e nome, codice persona**.
- Il tempo a disposizione è di 2 ore.
- Non separate questi fogli. Scrivete la soluzione solo sui fogli distribuiti, utilizzando il retro delle pagine in caso di necessità. Cancellate le parti di brutta con un tratto di penna.
- Ogni parte non cancellata a penna sarà considerata parte integrante della soluzione.
- È possibile scrivere a matita (e non occorre ricalcare al momento della consegna!).
- **È vietato utilizzare telefoni o pc**. Chi tenti di farlo vedrà annullata la sua prova.
- **Non è ammessa la consultazione di libri e appunti.**
- **Qualsiasi tentativo di comunicare con altri studenti comporta l'espulsione dall'aula**.
- È possibile ritirarsi senza penalità.
- **Non è possibile lasciare l'aula conservando il tema della prova in corso**.

Esercizio 1 (7 punti)

Implementare la seguente funzione:

```
void inverti(int v[], int n),
```

che riceve in ingresso un vettore `v` di lunghezza `n` e lo inverte, in modo che l'ultimo elemento di `v` diventi il primo, il penultimo diventi il secondo e così via. **Si implementi una funzione in grado di invertire il vettore `v` senza crearne una copia e senza utilizzare un ulteriore variabile array a supporto.**

Esempio: Il vettore {1,2,3,4,5} deve essere trasformato in {5,4,3,2,1}.

Soluzione

```
void inverti(int v[], int n)
{
  int i,tmp;
  for (i=0; i<n/2; i++)
  {
    tmp = v[i];
    v[i] = v[n-i-1];
    v[n-i-1] = tmp;
  }
}
```

Esercizio 2 (8 punti)

A) Si implementi in C il tipo di dato `alimento`, che consente di rappresentare i dati nutrizionali per 100g di uno specifico alimento: kcal, proteine (g), grassi (g), carboidrati (g).

B) Si implementi in C il tipo di dato `piatto`, che consente di rappresentare i dati di uno specifico piatto: i dati nutrizionale degli alimenti che lo compongono (non più di 50 diversi alimenti) e la loro quantità (in grammi).

C) Si implementi la funzione `float grassi(piatto p)`, che riceve in ingresso un piatto p e ritorna la percentuale di grassi che contiene (calcolata sulla base dei dati nutrizionali degli alimenti che contiene).

Soluzione

```
typedef struct
{


  int kcal;
  float proteine;
  float grassi;
  float carboidrati;



} alimento;

typedef struct
{


  alimento ingredienti[50];
  float peso[50];
  int ning;


} piatto;

float grassi(piatto p)
{









  int i;

  float pesoTot=0, grassi=0;

  for (i=0; i<p.ning; i++)
  {

    pesoTot = pesoTot + p.peso[i];

    grassi = grassi + p.ingredienti[i].grassi * p.peso[i]/100.0;

  }

  return grassi/pesoTot;









}
```

Esercizio 3 (8 punti)

Implementare in C la funzione ricorsiva `int trova_somma (int somma, int v[], int n)` che riceve in ingresso un vettore `v` di lunghezza `n` e restituisce 1 se il vettore `v` contiene uno o più elementi che sommati insieme hanno valore pari al parametro `somma`, viceversa restituisce 0.

Esempi

Sia `v={3,7,9,4,12}`, `trova_somma(13,v,5)` restituirà 1 (9+4=13), `trova_somma(31,v,5)` restituirà 0 (anche se 3+7+9+12=31, gli elementi non sono consecutivi), `trova_somma(19,v,5)` restituirà 1 (3+7+9=19).

Soluzione

```c
int trova_somma (int somma, int v[], int n)
{
  int i,ris;
  if (somma<0 || n==0)
    return 0;

  for (i=0; i<n; i++)
  {
    if (v[i]==somma)
      return 1;
    else
    {
      ris = trova_somma(somma-v[i], v+i+1, n-1-i);
      if (ris==1)
        return 1;
    }
  }
  return 0;
}
```

Esercizio 4 (4 punti)

Dati i due numeri A = −48 e B = -10, codificare entrambi in complemento a 2 (CPL2), utilizzando il numero minimo di bit necessari a rappresentare entrambi. Si effettuino quindi le operazioni A+B e A-B indicando esplicitamente se si verifica overflow o meno, e motivando la risposta. Mostrare i passaggi fatti.

Soluzione

Con 7 bit, in CPL2 possono essere rappresentati i numeri fra -64 e +63.

A = -48 → rappresento -48+128 = 80 = 64+16 → A = 1010000_{CPL2}
B = -10 → rappresento -10+128 = 118 = 64+32+16+4+2 → B = 1110110_{CPL2}

```
      11
A     1010000
+
B     1110110
----------------
   (1) 1000110
```

Non si verifica overflow, perché gli operandi hanno segno concorde con il risultato.

-B = +10 → 0001010_{CPL2}

```
A     1010000
-
B     0001010
----------------
      1011010
```

Non si verifica overflow, perché gli operandi hanno segno discorde.

Esercizio 5 (5 punti)

Si consideri il seguente programma C:

```c
int main()
{
  int x,y,v;

  scanf("%d",&x);
  scanf("%d",&y);
  scanf("%d",&v);
  while (v>0)
  {
    if (v>=x && v<=y)
      printf("%d\n",v);
      scanf("%d",&v);
  }
  return 0;
}
```

Implementare un programma in linguaggio macchina **equivalente** (cioè che produca lo stesso output a parità di input) al programma C qui sopra riportato.

Soluzione

```
01.      READ
02.      STORE 101
03.      READ
04.      STORE 102
05.      READ
06.      STORE 103
07.      BLE 16
08.      SUB 101
09.      BLE 15
10.      LOAD 103
11.      SUB 102
12.      BGE 15
13.      LOAD 103
14.      WRITE
15.      BR 5
16.      END
```