	Politecnico di Milano Scuola di Ingegneria Industriale e dell'Informazione **FONDAMENTI DI INFORMATICA** Prima prova in itinere - 25 Novembre 2016		COGNOME E NOME
	RIGA	COLONNA	MATRICOLA
			Spazio riservato ai docenti

- Il presente plico contiene 4 esercizi e **deve essere debitamente compilato con cognome e nome, numero di matricola**.
- Il tempo a disposizione è di 1 ore e 30 minuti.
- 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.
- 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 (5 punti)

```
#include <stdio.h>

int divisori(int num, int v[], int p[], int l);

int main()
{
  int num, divi[100], v[100], nd, nv;

  scanf("%d",&num);

  /* Inserire qui il codice richiesto dalla DOMANDA A */

  nd = divisori(num,v,divi,nv);

  int i=0;
  for (i=0; i<nd; i++)
    printf("%d ",divi[i]);
  printf("\n");

  return 0;
}
```

A. Scrivere un frammento di codice (da inserire dove indicato nello scheletro sopra riportato) che memorizzi in `v` una sequenza di numeri interi maggiori di 1 inseriti da tastiera; la sequenza termina non appena viene inserito un valore minore o uguale a 1, oppure al raggiungimento di 100 elementi maggiori di 1; la lunghezza della sequenza letta (ad eccezione dell'ultimo eventuale elemento non positivo) deve essere memorizzata nella variabile `nv`.

B. Implementare la funzioni `divisori` in modo che (i) la chiamata `nd=divisori(num,v,divi,nv)` ritorni il numero di elementi in `v` che sono divisori di `num`, e (ii) che l'array `divi` alla fine dell'esecuzione della funzione contenga (a partire dalla prima posizione) solo gli elementi di `v` che sono divisori di `num`.

Soluzione

```
#include <stdio.h>

int divisori(int num, int v[], int p[], int l);

int main(){
  int num, divi[100], v[100], nd, nv;

  scanf("%d",&num);
  nv = 0;
  do{
      scanf("%d",&v[nv]);
      nv++;
  } while(v[nv-1]>1);
  nv--;

  nd = divisori(num,v,divi,nv);

  int i=0;
  for (i=0; i<nd; i++)
    printf("%d ",divi[i]);
  printf("\n");

  return 0;
}

int divisori(int num, int v[], int p[], int l){
  int i,j=0;
  for (i=0; i<l; i++)
    if (num%v[i]==0){
      p[j] = v[i];
      j++;
    }
    return j;
}
```

Esercizio 2 (6 punti)

Il sistema informativo di una mensa gestisce i menu e le scelte dei clienti. Ogni giorno il menu contiene $\mathbf{n_1}$ scelte per la prima portata e $\mathbf{n_2}$ scelte per la seconda portata (dove $\mathbf{n_1}$ e $\mathbf{n_2}$ sono comprese fra 2 e 5). Per ogni portata occorre memorizzare il nome della portata (una stringa lunga al più 100 caratteri) e se si tratta di una portata vegetariana. All'inizio della settimana, ciascun cliente deve scegliere per, ogni giorno della settimana, una prima portata e una seconda portata fra quelle a disposizione. Per ogni giorno e per ogni portata, la scelta viene rappresentata con un numero, compreso fra 1 e $\mathbf{n_i}$ (dove $\mathbf{n_i}$ è il numero di scelte disponibili per quella portata nel menu del giorno).

A. Definire un tipo di dato `portata` che contiene tutte le informazioni riguardanti una portata (il nome e se è vegetariana)
B. Usando il tipo `portata` appena definito, definire il tipo di dato `pasto` che contiene tutte le informazioni sul menu del giorno (numero di scelte per ciascuna portata e le informazioni per ciascuna portata che può essere scelta).

Si considerino le seguenti dichiarazioni di variabili, dove l'array `settimana` contiene tutti i menu della settimana corrente (dal lunedì al venerdì); la matrice `menu` contiene le portate scelte dal cliente della mensa; ogni riga contiene la scelta della prima portata (prima colonna) e della seconda portata (seconda colonna) di ciascun giorno.

```
pasto settimana[5];
int menu[5][2];
```

C. Ipotizzando che `settimana` e `menu` siano già stati opportunamente inizializzati, scrivere un frammento di programma che prima di tutto verifichi che le scelte effettuate siano valide (cioè ogni portata scelta sia compresa fra 1 e il numero di portate previste per quel giorno e quel tipo di portata); in seguito stampi i nomi delle portate scelte (stampare una riga per ogni giorno della settimana con le due portate separate da una virgola); infine stampi se il menu è vegetariano (cioè se tutte le scelte sono portate vegetariane).

Soluzione

```
#include <stdio.h>

typedef struct
{
  char nome[100];
  enum {si,no} veg;
} portata;

typedef struct
{
  portata primi[5];
  portata secondi[5];
  int n1;
  int n2;
} pasto;

int main()
{
  pasto settimana[5];
  int menu[5][2];

  /* Inizializazzione di settimana e di scelte */

  int veg=1, corretto=1, i,j;

  for (i=0; i<5 && corretto==1; i++)
  {
      int primo = menu[i][0];
      int secondo = menu[i][1];

      if (primo<1 || primo>settimana[i].n1 || secondo<1 || secondo>settimana[i].n2)
        corretto = 0;
  }

  if (corretto==0)
    printf("Il menu non è stato scelto correttamente\n");
  else
  {
    for (i=0; i<5 && corretto==1; i++)
    {
        int primo = menu[i][0];
        int secondo = menu [i][1];

        printf ("%s,%s\n",settimana[i].primi[primo-1].nome,
                                  settimana[i].secondi[secondo-1].nome);

        if (settimana[i].primi[primo-1].veg == no ||
                  settimana[i].secondi[secondo-1].veg == no)
            veg = 0;
    }

    if (veg == 1)
      printf ("Il menu della settimana è vegetariano\n");
    else
      printf ("Il menu della settimana non è vegetariano\n");
    }

  return 0;
}
```

Esercizio 3 (3 punti)

Rispondere alle seguenti domande negli spazi indicati. Nello spazio sottostante, **riportare i calcoli e/o le giustificazioni per tutte le risposte date**.

A. Qual è il numero minimo di bit che permette di codificare **tutti** i seguenti numeri in complemento a due: -32,+6,-6?
Risposta: _____ bit

B. Usando il numero di bit individuato al punto precedente, codificare in complemento a 2 i seguenti valori:

- -32_{10} = _____ CPL2
- $+6_{10}$ = _____ CPL2
- -6_{10} = _____ CPL2

C. Eseguire le somme in complemento a 2: (i) -32+6 e (ii) -32-6 negli spazi qui sotto:

(-32_{10}) + $(+6_{10})$ ---------- = Si è verificato overflow? ☐ SI ☐ NO	(-32_{10}) + (-6_{10}) ---------- = Si è verificato overflow? ☐ SI ☐ NO

Giustificazioni e calcoli

Con n bit si possono rappresentare in complemento a 2 i numeri compresi fra -2^{n-1} fino a $2^{n-1} - 1$
In questo caso, il numero che richiede più bit per essere rappresentato è -32, che ne richiede 6 → devo usare 6bit

$-32_{10} = 100000_{CPL2}$ (si ottiene rappresentando 64-32 = +32)
$+6_{10} = 000110_{CPL2}$ (si ottiene senza trasformazione)
$-6_{10} = 111010_{CPL2}$ (si ottiene rappresentando 64-6 = 58)

Eseguo le somme:

```
   100000   (-32₁₀)
   000110   (+6₁₀)
----------------
(1)100110   (-26₁₀)
```

Non si può verificare overflow perché gli operandi hanno segno discorde.

```
   100000   (-32₁₀)
   111010   (-6₁₀)
----------------
(1)011010   (OVERFLOW)
```

Si verifica overflow perché il risultato non ha segno concorde con gli operandi

Esercizio 4 (2 punti)

Si consideri la seguente espressione booleana:

E = (A AND NOT B) OR (C AND NOT A)

Si compili la seguente tabella della verità (in cui 0 rappresenta il valore logico FALSO, 1 il valore VERO):

A	*B*	*C*	*NOT B*	*A AND NOT B*	*NOT A*	*C AND NOT A*	*E*
0	0	0					
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1					

Si consideri ora la condizione, scritta in linguaggio C, in cui x e y siano due variabili float:

((y>3) && !(y<0)) || ((x>5) && !(y>3))

ottenuta dalla prima formula sostituendo la variabile A con y>3, la variabile B con y<0, la variabile C con x>5

Per quali valori di x e y, la condizione risulta FALSA?

Soluzione

A	*B*	*C*	*NOT B*	*A AND NOT B*	*NOT A*	*C AND NOT A*	*E*
0	0	0	1	0	1	0	0
0	0	1	1	0	1	1	1
0	1	0	0	0	1	0	0
0	1	1	0	0	1	1	1
1	0	0	1	1	0	0	1
1	0	1	1	1	0	0	1
1	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0

Effettuate le sostituzioni di A, B e C proposte, si osserva che:
per y<0 → A=0 e B=1
per 0<=y<=3 → A=0 e B=0
per y>3 →A=1 e B=0

Quindi per nessun valore di y è possibile avere A=1 e B=1. Le ultime due righe della tabella di verità non si possono verificare. Pertanto, la condizione è FALSA in corrispondenza della riga 1 e 3, cioè quando C=0 e quando A=0 (infatti B può essere sia 0 che 1). Quindi la condizione è FALSA se y<=3 e x<=5.