LEGGETE LA GUIDA PER LA CREAZIONE DEI PROGETTI E PER IL DEBUGGING!

Gli esercizi seguenti devono essere risolti, compilati e testati utilizzando il debugger. Per ognuno si deve realizzare una funzione main() che ne testi il funzionamento. ***Fate progetti diversi per ogni esercizio.***

Esercizio 1

Sia dato il seguente programma C, nel file "main.c":

```c
#include <stdlib.h>
#include <stdint.h>

/* Scrivere qui la funzione raddoppia_tre() */

int main(void)
{
    uint32_t *v = malloc(3 * sizeof(uint32_t));
    v[0] = 12;
    v[1] = 59;
    v[2] = 83;
    raddoppia_tre(v);
    free(v);
    return 0;
}
```

Scrivere la definizione della funzione `raddoppia_tre()` sapendo che, dopo la sua esecuzione, i tre valori presenti nel vettore di interi senza segno a 32 bit devono essere raddoppiati.

Esercizio 2

Sia dato il seguente programma C, nel file "main.c":

```c
#include <stdlib.h>
#include <stdint.h>

void raddoppia(uint32_t *x, size_t n)
{
    /* completare la funzione raddoppia() */
}

int main(void)
{
    size_t n = 3;
    uint32_t *v = malloc(n * sizeof(uint32_t));
    v[0] = 12;
    v[1] = 59;
    v[2] = 83;
    raddoppia(v, n);
    free(v);
    return 0;
}
```

Riempire la definizione della funzione `raddoppia()` sapendo che deve prendere un vettore di interi senza segno a 32 bit e il loro numero `n`, e per ognuno dei suoi elementi deve calcolarne il doppio e memorizzarlo all'interno del vettore nella stessa posizione.

Una volta realizzata e testata la funzione, modificare n (portandolo ad esempio a 5) e aggiungere nel vettore altri valori. Verificare che la funzione continui a produrre il risultato atteso.

Esercizio 3

Creare un file “main.c”. Nel file, si realizzi in linguaggio C la funzione corrispondente alla seguente dichiarazione:

```
extern void iota(uint32_t *x, size_t n);
```

La funzione deve riempire il vettore x di n interi senza segno a 32 bit con il relativo indice. Ad esempio: il primo dovrà essere inizializzato a 0, il secondo a 1, il terzo a 2 e così via.

Esercizio 4

Creare un file “main.c”. Nel file, si realizzi in linguaggio C la funzione corrispondente alla seguente dichiarazione:

```
extern void copy(uint32_t *dst, uint32_t *src, size_t n);
```

La funzione riceve come parametri:

- `dst` un puntatore destinazione ad una zona di memoria in grado di contenere n interi senza segno a 32 bit
- `src` un puntatore sorgente ad una zona di memoria in grado di contenere n interi senza segno a 32 bit
- `n` il numero di interi che descrive la dimensione delle due zone di memoria.

Il suo scopo è quello di copiare gli n interi puntati da `src` in `dst`.

Esercizio 5

Creare un file “main.c”. Nel file, si realizzi in linguaggio C la funzione corrispondente alla seguente dichiarazione:

```
extern uint32_t *somme_2a2(uint32_t *vett, size_t size);
```

La funzione `somme_2a2()` deve restituire un vettore allocato dinamicamente di `size/2` interi senza segno a 32 bit che conterranno le somme due a due dei valori contenuti nel vettore passato come parametro, ovvero nel primo valore del vettore `risultante` dovrà esserci la somma del primo e del secondo valore di `vett`, nel secondo la somma del terzo e del quarto valore di `vett`, e così via. Se `size` è dispari la funzione somme_2a2 salta l’ultimo valore del vettore passato.

Un esempio di `main()` che utilizza la funzione è il seguente:

```
int main(void)
{
    size_t n = 6;
    uint32_t *v = malloc(n * sizeof(uint32_t));
    v[0] = 3; v[1] = 87; v[2] = 5; v[3] = 7; v[4] = 12; v[5] = 9;
    uint32_t *somme = somme_2a2(v, n);
    free(v);
    free(somme);
    return 0;
}
```

La variabile somme punterà a 3 interi a 32 bit senza segno del valore rispettivamente di 90, 12, e 21.
Liberate sempre la memoria quando testate i programmi, perché la free effettua dei controlli che possono segnalare eventuali problemi nel vostro codice.

Esercizio 6

Creare un file “main.c”. Nel file, si realizzino in linguaggio C le funzioni corrispondenti alle seguenti dichiarazioni:

```
extern double media (double *x, unsigned int n);
extern double varianza (double *x, unsigned int n);
```

Le funzioni devono calcolare media e varianza (cercate la formula su Wikipedia se non la ricordate) del vettore x contenente n elementi. Se n è 0 la media è zero, se n è minore di due la varianza è zero.

Nel testare il programma si scriva una funzione `main()` in cui il vettore deve essere allocato dinamicamente utilizzando le funzioni `malloc()` e `free()`.

Esercizio 7

Spostare le funzioni dell’esercizio precedente in un altro file C chiamato “stat.c”, creare il corrispondente header “stat.h” (che conterrà solo le dichiarazioni) e includerlo nel file “main.c”. Ricordate di inserire nel file .h l’include guard e nel file .c di includere l’header corrispondente. Verificare il funzionamento e il comportamento col debugger.

Esercizio 8

Nel file `calcola_seno.c` implementare la definizione della funzione:

```
extern double calcola_seno(double x);
```

La funzione deve calcolare il valore di $\operatorname{sen} x$ utilizzando la seguente equazione:

$$\operatorname{sen} x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1}$$

Il parametro x è un angolo espresso in radianti. Si ottiene una precisione sufficiente con almeno 20 iterazioni. Farne poi una versione in cui il numero di iterazioni non sia fissato a priori, ma si fermi una volta che il risultato non cambia più.