Laboratorio di Calcolo per Fisici,
Esame del 07/02/2025
A.A. 2024/2025

Nome _____ Cognome _____

Matricola _____ ☐ Ritirato/a

Lo scopo di questa esercitazione è di scrivere un programma in C e uno script in python seguendo la traccia riportata di seguito. Si tenga presente che:

1. Per svolgere il compito si hanno a disposizione 3 ore.
2. Si possono usare libri di testo, prontuari e gli appunti ma non è ammesso parlare con nessuno né utilizzare cellulari, tablet o laptop, pena l'annullamento del compito.
3. Il programma va scritto e salvato esclusivamente sul computer del laboratorio.
4. **Tutti i file vanno salvati in una cartella chiamata `ELCFEB_NOME_COGNOME` nella home directory**, dove `NOME` e `COGNOME` indicano rispettivamente il tuo nome e cognome. Ad esempio lo studente *Marco Rossi* deve creare una cartella chiamata `ELCFEB_MARCO_ROSSI` contenente tutti i file specificati nel testo. **Tutto ciò che non si trova all'interno della cartella suddetta non verrà valutato.** In tutti i programmi e script inserisci all'inizio un commento con il tuo nome, cognome e numero di matricola.
5. **Dovete consegnare il presente testo indicando nome, cognome e numero di matricola** (vedi sopra), barrando la casella "Ritirato/a" se ci si vuole ritirare, ovvero se non si vuole che la presente prova venga valutata.
6. **Per consegnare il compito** dovrete eseguire, all'interno della cartella creata in precedenza (come spiegato al punto 4), il seguente comando da terminale: `cp * /media/sf_esame/`

Si consideri un sistema monetario con un insieme di M tagli di monete disponibili, ad esempio $V = \{1, 2, 3, 5\}$. Dato un valore intero positivo x, si vuole determinare il numero minimo di monete necessarie per ottenere x usando i tagli disponibili.
Per risolvere il problema si può usare un approccio iterativo basato sulla relazione seguente:

$$m(x) = \begin{cases} \infty & \text{per } x < 0 \\ 0 & \text{per } x = 0 \\ \min_{i \in V}\{m(x - V_i)\} + 1 & \text{altrimenti} \end{cases} \quad (1)$$

dove $m(x)$ rappresenta il numero minimo di monete per ottenere il valore x, e V_i è l'i-esimo taglio presente nell'insieme V (cioè se V={1,2,3,5}, $V_0 = 1$, $V_1 = 2$, $V_2 = 3$ e $V_3 = 5$). Come si può vedere, la soluzione per un dato x, cioè $m(x)$, viene espressa in termini di soluzioni $m(z)$, con $z < x$ (dove z assume i valori $x - V_0$, $x - V_1$ ... $x - V_{M-1}$). Questo significa che per trovare il numero minimo di monete necessarie per ottenere x è sufficiente immagazzinare in un array il numero minimo di monete necessarie per ottenere tutti i valori minori di x. Un algoritmo efficiente per fare queste operazione è il seguente:

- Definire un array `m[XMAX + 1]`, dove `XMAX` è l'ammontare massimo che si vuole poter cambiare, mettendo a 0 il primo elemento (cioè $m[0] = 0$).
- Definire una macro `INF` per rappresentare un valore molto grande (esempio: 100000), da usare al posto di ∞.
- Fare un ciclo per y che va da 1 a x (estremi compresi); ad ogni iterazione applicare l'eq. (1) per ottenere $m(y)$, immagazzinando il suo valore in `m[y]`. Notando che in tale ciclo in y l'eq. (1) si applica solo per valori di y positivi, per calcolare $m(y)$ sarà sufficiente utilizzare un ciclo di M iterazioni per determinare il minimo tra i valori appartenenti all'insieme di M elementi $\{m(y - V_0), \ldots m(y - V_{M-1})\}$, considerando che, se $z = y - V_i$ (con $i = 0 \ldots M - 1$) è minore di 0 allora $m(z) = \infty$ (ovvero `m[z]=INF` nel vostro codice C), altrimenti si dovrà utilizzare il valore di m già calcolato in precedenza, ovvero $m[z]$.

Alla fine dell'algoritmo ogni elemento y dell'array `m[]` (compreso l'elemento x-esimo) conterrà il numero minimo di monete in V necessarie per ottenere y, e quindi anche x, che è l'ammontare maggiore.
Considerate come esempio il caso $V = \{1, 3\}$ e $x = 7$, che potete utilizzare per testare il vostro codice. Applicando l'algoritmo presentato sopra, i valori di $m(y)$, per $y \leq x$, si calcolano iterativamente come segue:

- $m(0) = 0$.
- $m(1) = 1$.
- $m(2) = \min\{m(2-1), m(2-3)\} + 1 = \min\{m(1), \infty\} + 1 = \min\{1, \infty\} + 1 = 2$.
- $m(3) = \min\{m(3-1), m(3-3)\} + 1 = \min\{m(1), m(0)\} + 1 = 1$.
- $m(4) = \min\{m(4-1), m(4-3)\} + 1 = \min\{m(3), m(1)\} + 1 = 2$.
- $m(5) = \min\{m(5-1), m(5-3)\} + 1 = \min\{m(4), m(2)\} + 1 = 3$.
- $m(6) = \min\{m(6-1), m(6-3)\} + 1 = \min\{m(5), m(3)\} + 1 = 2$.
- $m(7) = \min\{m(7-1), m(7-3)\} + 1 = \min\{m(6), m(4)\} + 1 = 3$.

► **Prima parte:**
Si realizzi un programma in C, chiamato `nome_cognome.c` (tutto minuscolo, senza eventuali spazi, accenti o apostrofi), che calcoli e stampi *tutti* i valori $m(y)$ con $0 \leq y \leq x$ per tre insiemi di $M = 4$ tagli di monete ciascuno, $V_1 = \{1, 2, 3, 4\}$, $V_2 = \{1, 2, 3, 5\}$, e $V_3 = \{1, 2, 3, 6\}$ e $x \leq$ `XMAX` dato in input dall'utente.
In particolare il programma dovrà:

1. Chiedere all'utente di inserire un valore $x \in [2,$ `XMAX`$]$, con `XMAX = 100`.
2. Calcolare $m(y)$ con $0 \leq y \leq x$, immagazzinando i risultati in un array di interi `m[]`, per ognuno dei tre set $V^{(1)}$, $V^{(2)}$ e $V^{(3)}$. **Suggerimento**: usate un array `V[3][4]` per immagazzinare i tre set e servitevi di un ciclo su $j = 0, 1, 2$ per "scorrerli".
3. Stampare y e $m(y)$ su tre file, uno per ogni set, usando la funzione `stampa_m()` (vedi sotto per le specifiche di tale funzione).
4. Definire i valori di `M`, `XMAX` e `INF` con delle opportune macro.

Nello scrivere il programma si richiede che vengano implementate almeno le seguenti funzioni:

- `inserisci_x()` che chiede all'utente il valore di x da valutare, reiterando la richiesta qualora il valore inserito non sia compreso tra 2 e `XMAX`, e lo restituisce tramite un puntatore a intero passato come argomento.
- `trova_min()` che prende come parametri un valore x, l'array `m[]` e quello contenente il set di tagli da utilizzare (cioè $V^{(j)}$ dove $j = 0, 1, 2$), e restituisce $m(x)$ applicando l'equazione (1) come spiegato in precedenza.
- `stampa_m()` che prende in input l'array `m[]`, un intero `i` ed il valore x e stampa su file i valori `y m[y]` per ogni y tra 0 e x compresi (ovvero in ogni riga di tale file si deve avere y e il corrispondente valore $m(y)$); il file deve chiamarsi "min_i.dat", dove `i` è il valore del parametro in input e rappresenta il set i-esimo dei tagli da utilizzare. **Suggerimento:** potete usare `sprintf()` per scrivere su una stringa.

Suggerimento: prima di considerare il caso generale con $M = 4$ e l'array 2D `V[3][4]`, implementate il caso mostrato nell'esempio con $M = 2$ e $V = \{1, 3\}$ per testare il vostro codice.

► **Seconda parte:** Utilizzando i tre file generati eseguendo il programma scritto nella prima parte con $x = 91$, creare uno script `python` chiamato `nome_cognome.py` che crei un grafico che mostri $m(y)$ in funzione di y per i tre set di tagli di monete, configurando il grafico in modo che abbia legenda ed etichette appropriate. Lo script deve salvare il grafico su un file chiamato `min.png`.