1506
UNIVERSITÀ
DEGLI STUDI
DI URBINO
CARLO BO

UNIVERSITÀ DEGLI STUDI DI URBINO CARLO BO
DIPARTIMENTO DI SCIENZE PURE E APPLICATE
CORSO DI LAUREA DI INFORMATICA APPLICATA

Progetto di Programmazione Logica e Funzionale

Sessione Autunnale 2022/2023

Professore Marco Bernardo

Marzio Della Bosca
Matricola: 306034

Susanna Peretti
Matricola: 306186

Indice

1
Specifica del Problema

Scrivere un programma Haskell e un programma Prolog che acquisiscono da tastiera una lista finita di numeri reali che rappresenta un segnale nel dominio del tempo e applicano al segnale la Trasformata discreta di Fourier (DFT). Successivamente richiedono la tipologia di filtro (passa-basso, passa-alto o passa-banda) da utilizzare, la\e frequenza\e di taglio ed eseguono il filtraggio nel dominio della frequenza. Infine, applicano la Trasformata discreta di Fourier Inversa (IDFT) e stampano il segnale filtrato nel dominio del tempo.

2
Analisi del Problema

2.1 Dati di Ingresso del Problema

I dati di ingresso del problema sono rappresentati da una lista finita di numeri reali rappresentante il segnale nel dominio del tempo, un numero reale positivo che rappresenta la frequenza di campionamento del segnale e un numero reale positivo rappresentante la frequenza di taglio che il filtro andrà ad utilizzare. Nel caso venga utilizzato il filtro passa banda, le frequenze di taglio saranno due così come i numeri reali positivi in ingresso.

2.2 Dati di Uscita del Problema

L'unico dato in uscita del problema è una lista finita di numeri reali rappresentante il segnale filtrato nel dominio del tempo.

2.3 Relazioni Intercorrenti tra i Dati del Problema

La Trasformata Discreta di Fourier (DFT) è un'operazione matematica che consente di analizzare un segnale, una sequenza finita e ordinata di dati, nel dominio delle frequenze. Per applicare la DFT a una lista finita di numeri reali, è necessario convertire questi numeri in numeri complessi. Ad esempio, il numero reale x può essere rappresentato come il numero complesso $x + 0j$, dove j è l'unità immaginaria.

2.3.1 Trasformata Discreta di Fourier (DFT)

La sequenza finita di numeri complessi di lunghezza $N \in \mathbb{N}$

$$x_0,\, x_1,\, \ldots\, x_{n-1}$$

è trasformata nella sequenza finita di $N \in \mathbb{N}$ numeri complessi

$$X_0,\, X_1,\, \ldots\, X_{n-1}$$

tramite la Trasformata Discreta di Fourier secondo la seguente formula:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi}{N}kn}$$

dove:

- $-j\frac{2\pi}{N}$ è una radice dell'unità primitiva N-esima

- N è la dimensione del segnale
- $X[k]$ è il valore nel dominio della frequenza dell'indice k
- $x[n]$ è il campione del segnale, nel dominio del tempo, all'indice n
- j è l'unità immaginaria, ovvero $\sqrt{-1}$.

2.3.2 Filtri

Un filtro, nella teoria dei segnali, è un sistema utilizzato per modificare le caratteristiche di un segnale attraverso l'annullamento o l'attenuazione delle ampiezze di particolari frequenze. Dato un segnale, nel dominio della frequenza, la frequenza del k-esimo elemento è data dalla formula:

$$f(k) = (k/N) * Fc$$

dove:

- k è l'indice del k-esimo elemento
- N è la dimensione del segnale
- Fc è la frequenza di campionamento del segnale

Si considerano tre tipologie di filtro:

- un filtro passa basso consente il passaggio delle frequenze inferiori ad una certa frequenza di taglio e abbatte l'ampiezza alle frequenze superiori;
- un filtro passa alto lascia passare le frequenze superiori ad una certa frequenza di taglio e abbatte l'ampiezza alle frequenze inferiori;
- un filtro passa banda permette il passaggio di frequenze all'interno di un dato intervallo, definito come banda passante, e abbatte le ampiezze alle frequenze maggiori della frequenza massima di taglio e minori della frequenza minima di taglio.

2.3.3 Trasformata Discreta di Fourier Inversa (IDFT)

La Trasformata Discreta di Fourier Inversa è il processo opposto alla DFT. Preso un segnale nel dominio della frequenza, restituisce il segnale nel dominio del tempo secondo la seguente formula:

$$X[n] = \frac{1}{N} \sum_{n=0}^{N-1} x[k] \cdot e^{j\frac{2\pi}{N}kn}$$

- $j\frac{2\pi}{N}$ è una radice dell'unità primitiva N-esima
- N è la dimensione del segnale
- $X[n]$ è il valore nel dominio del tempo dell'indice n
- $x[k]$ è il campione del segnale, nel dominio della frequenza, all'indice k
- j è l'unità immaginaria, ovvero $\sqrt{-1}$.

3
Progettazione dell'Algoritmo

3.1 Scelte di Progetto

I segnali nel dominio del tempo si prestano in modo naturale a essere rappresentati mediante strutture dati lineari, quindi sotto forma di liste finite di numeri reali.

La scelta di utilizzare il tipo *Double*, nella rappresentazione degli elementi dei segnali, per i valori delle frequenze di taglio e di campionamento, anziché il tipo *Float*, è motivata dalla maggiore precisione offerta dal tipo *Double*, grazie ai suoi 64 bit rispetto ai 32 bit di *Float*. Questo è utile per preservare dettagli critici nei segnali nel dominio del tempo e delle frequenze, evitando errori di arrotondamento che potrebbero compromettere l'accuratezza delle analisi e delle operazioni matematiche.

È stato deciso di eseguire il filtraggio del segnale nel dominio delle frequenze, escludendo il primo elemento, poiché spesso questo elemento rappresenta informazioni di rilievo correlate agli altri componenti del segnale e, inoltre, è solitamente il più energetico. Di conseguenza il filtraggio di questo primo componente potrebbe causare una perdita di qualità e di informazioni del segnale stesso.

Infine, è stato deciso di evitare di chiedere all'utente l'immissione delle frequenze di taglio nel caso in cui il segnale in input sia rappresentato da una lista vuota, in quanto definire una frequenza di taglio anziché un'altra non avrebbe nessun effetto.

3.2 Passi dell'Algoritmo

I passi dell'algoritmo per risolvere il problema sono i seguenti:

- Acquisire il segnale nel dominio del tempo rappresentato da una lista finita di numeri reali.
  - Se non si inserisce una lista di numeri reali, viene richiesto il reinserimento.
- Acquisire il valore rappresentante la frequenza con cui è stato campionato il segnale.
  - Se il valore inserito non è un numero reale positivo, viene richiesto il reinserimento.
- Convertire la lista di numeri reali summenzionata in una lista di numeri complessi:
  - Caso base: se la lista è vuota restituisce una lista vuota.
  - Caso generale: se la lista non è vuota si prende il primo elemento della lista come parte reale e si pone la parte immaginaria a 0, poi si procede ricorsivamente con il resto della lista.
- Applicare la trasformata discreta di Fourier al segnale:

  - Caso base: se la lista è vuota, restituisce una lista vuota.
  - Caso generale: se la lista non è vuota, viene calcolato il k-esimo elemento della DFT della lista, quindi la funzione procede ricorsivamente attraverso la lista calcolando la sommatoria di ogni elemento della lista per l'esponente di un numero complesso definito come $e^{-j\frac{2\pi}{N}kn}$, dove n è l'indice dell'elemento della lista in input, k è l'indice dell'elemento nella DFT, e N rappresenta la dimensione totale della lista. La somma di tutti questi prodotti è l'elemento k-esimo della DFT del segnale.

- Stampare il segnale summenzionato nel dominio della frequenza.
- Acquisire la scelta relativa al filtro da applicare.
  - Se il valore inserito non è compreso tra quelli definiti per la scelta del filtro, viene richiesto il reinserimento.
- Acquisire il\i valore\i della\e frequenza\e di taglio.
  - I valori inseriti per le frequenze devono essere numeri reali positivi, in caso contrario si richiede il reinserimento.
  - Nel caso si scelga il filtro passa banda i valori delle frequenze di taglio richieste sono due, di cui la frequenza di taglio minore deve essere strettamente minore alla frequenza di taglio superiore. In caso contrario si richiede il reinserimento di entrambe le frequenze.
- Applicare il filtro al segnale:
  - Filtro passa basso:
    * Caso base: se la lista è vuota restituisce una lista vuota.
    * Caso generale: se la lista non è vuota, viene calcolata la frequenza del k-esimo elemento a partire dal secondo elemento, se è inferiore alla frequenza di taglio si passa all'elemento successivo, in caso contrario la parte reale viene posta a 0 e si procede ricorsivamente con il resto della lista.
  - Filtro passa alto:
    * Caso base: se la lista è vuota restituisce una lista vuota.
    * Caso generale: se la lista non è vuota, viene calcolata la frequenza del k-esimo elemento a partire dal secondo elemento, se è inferiore alla frequenza di taglio la parte reale viene posta a 0 e si procede ricorsivamente fino a che il calcolo della frequenza del k-esimo elemento non sia maggiore o uguale alla frequenza di taglio, a questo punto la restante coda della lista viene restituita senza ulteriori operazioni.
  - Filtro passa banda:
    * Caso base: se la lista è vuota restituisce una lista vuota.
    * Caso generale: se la lista non è vuota si applica il filtro passa basso a cui come parametro viene passata la frequenza di taglio superiore e poi si applica il filtro passa alto a cui viene passata la frequenza di taglio inferiore.
- Stampare il segnale filtrato nel dominio della frequenza.
- Applicare la trasformata discreta di Fourier inversa:
  - Caso base: se la lista è vuota restituisce una lista vuota.

  - Caso generale: se la lista non è vuota viene calcolato il k-esimo elemento della IDFT della lista, quindi la funzione procede ricorsivamente attraverso la lista calcolando la sommatoria di ogni elemento della lista per l'esponente di un numero complesso definito come $e^{j\frac{2\pi}{N}kn}$, dove n è l'indice dell'elemento della lista in input, k è l'indice dell'elemento nella IDFT, e N è la dimensione della lista. La somma di tutti questi prodotti è l'elemento k-esimo della IDFT del segnale. Infine, ogni k-esimo elemento della lista così ottenuta viene poi normalizzato dividendolo per la dimensione della lista.

- Convertire la lista IDFT di numeri complessi summenzionata in una lista di numeri reali:
  - Caso base: se la lista è vuota restituisce una lista vuota.
  - Caso generale: se la lista non è vuota si prende la parte reale del primo elemento tralasciando la parte immaginaria, poi si procede ricorsivamente con il resto della lista.
- Stampare il segnale filtrato nel dominio del tempo.

4
Implementazione dell'Algoritmo

4.1 Programma Haskell

File sorgente: DFT_Filter_IDFT.hs

```
{- Programma Haskell per calcolare la trasformata discreta di Fourier (DFT) di un segnale,
   applicare un filtro e, infine, calcolare la trasformata discreta di Fourier inversa (IDFT)
   del segnale filtrato. -}

{- Importazione delle librerie. -}

{- Libreria necessaria per:
   - utilizzo della funzione readMaybe, legge una stringa in input, se la stringa può essere
     interpretata come il valore richiesto restituisce Just, se l'interpretazione non
     ha successo restituisce Nothing. -}
import Text.Read (readMaybe)

{- Libreria necessaria per:
   - rappresentazione di numeri complessi con coefficienti reali;
   - utilizzo della funzione realPart, restituisce la parte reale di un numero complesso;
   - utilizzo della funzione imagPart, restituisce la parte immaginaria di un numero complesso;
   - utilizzo dell'operatore ':+', utilizzato per la creazione di numeri complessi. -}
import Data.Complex

{- Main -}
{- Il programma accetta in input una lista finita di numeri reali rappresentante un segnale
   nel dominio del tempo e la frequenza di campionamento del segnale stesso, converte la
   lista di numeri reali in una lista di numeri complessi, calcola e stampa la trasformata
   discreta di Fourier (DFT) del segnale.
   Successivamente richiede la tipologia di filtro (passa basso, passa alto o passa banda)
   da applicare al segnale nel dominio delle frequenze e la/e frequenza/e di taglio.
   Dopo aver stampato il segnale filtrato nel dominio della frequenza, calcola la trasformata
   discreta inversa di Fourier (IDFT) del segnale e infine stampa il segnale ottenuto nel dominio
   del tempo. -}

main :: IO()
main = do
  putStrLn "Progetto per il corso di Programmazione Logica e Funzionale."
  putStrLn "Sessione Autunnale 2022/2023."
  putStrLn "Studente: Marzio Della Bosca, Matricola: 306034."
  putStrLn "Studente: Susanna Peretti,  Matricola: 306186."

  putStrLn "\n\nInserire il segnale da filtrare (es. [1, 2.3, -2]): "
  input_signal <- read_input_signal

  putStrLn "\nInserire la frequenza di campionamento del segnale: "
  sampling_rate <- read_double_value

  putStrLn "\nApplicazione della trasformata discreta di Fourier (DFT)."
  let dft_signal = dft (real_to_complex input_signal)
  putStrLn "Segnale nel dominio della frequenza:"
  putStrLn $ show (dft_signal)

  putStrLn "\nSelezione del filtro da applicare al segnale:"
  putStrLn " 1 - Filtro passa basso;"
  putStrLn " 2 - Filtro passa alto;"
```

```
  putStrLn " 3 - Filtro passa banda."
  putStrLn "Digitare 1, 2 o 3: "
  filter <- read_filter_choice

  filtered_signal <- filtering filter sampling_rate dft_signal

  putStrLn "\nSegnale filtrato nel dominio della frequenza:"
  putStrLn $ show (filtered_signal)

  putStrLn "\nApplicazione della trasformata discreta di Fourier inversa (IDFT)."
  let time_domain_filtered_signal = complex_to_real (idft filtered_signal)
  putStrLn "Segnale filtrato nel dominio del tempo:"
  putStrLn $ show (time_domain_filtered_signal)

{- Definizione delle funzioni -}

{- Definizione delle funzioni principali -}

{- Funzione per applicare la trasformata discreta di Fourier (DFT):
   - il suo unico argomento è il segnale nel dominio del tempo. -}
dft :: [Complex Double] -> [Complex Double]
dft [] = []
dft signal = dft_k_handler signal (length signal) 0

{- Funzione per la gestione dell'indice k per il calcolo della DFT:
   - il primo argomento è il segnale nel dominio del tempo;
   - il secondo argomento è la dimensione del segnale;
   - il terzo argomento è l'indice k, indica l'elemento a cui si sta facendo riferimento. -}
dft_k_handler :: [Complex Double] -> Int -> Int -> [Complex Double]
dft_k_handler [] _ _ = []
dft_k_handler signal size k
  | k == size = []
  | otherwise = (dft_k_sum signal size k 0) :
    (dft_k_handler signal size (k + 1))

{- Funzione per effettuare il calcolo del k-esimo elemento della DFT:
   - il primo argomento è il segnale nel dominio del tempo;
   - il secondo argomento è la dimensione del segnale;
   - il terzo argomento è l'indice k, indica l'elemento del segnale nel dominio della frequenza
     a cui si sta facendo riferimento;
   - il quarto argomento è l'indice n, indica l'elemento del segnale nel dominio del tempo
     a cui si sta facendo riferimento. -}
dft_k_sum :: [Complex Double] -> Int -> Int -> Int -> Complex Double
dft_k_sum [] _ _ _ = 0.0
dft_k_sum (x:xs) size k n =
  x * (cos esp :+ sin esp) + dft_k_sum xs size k (n + 1)
  where
    esp = - 2.0 * pi * fromIntegral n * fromIntegral k / fromIntegral size

{- Funzione per acquisire e validare il-i valore-i della-e frequenza-e di taglio
  ed eseguire l'applicazione del filtro scelto:
  - il primo argomento rappresenta il filtro selezionato;
  - il secondo argomento è la frequenza di campionamento del segnale;
  - il terzo argomento è il segnale da filtrare. -}
filtering :: Int -> Double -> [Complex Double] -> IO [Complex Double]
filtering _ _ [] = return []
filtering 1 sampling_rate signal = do
  putStrLn "\nInserire la frequenza di taglio per il filtro passa basso:"
  cutoff <- read_double_value
  return (low_pass_filter cutoff sampling_rate signal)
filtering 2 sampling_rate signal = do
  putStrLn "\nInserire la frequenza di taglio per il filtro passa alto:"
  cutoff <- read_double_value
  return (high_pass_filter cutoff sampling_rate signal)
filtering 3 sampling_rate signal = do
  putStrLn "\nInserire la frequenza di taglio inferiore:"
  cutoff_min <- read_double_value
  putStrLn "\nInserire la frequenza di taglio superiore:"
  cutoff_max <- read_double_value
  if cutoff_min >= cutoff_max
    then do
      putStr "Errore, la frequenza di taglio inferiore non deve essere "
      putStrLn "maggiore o uguale della frequenza di taglio superiore!"
```

```
      filtering 3 sampling_rate signal
    else return (band_pass_filter cutoff_min cutoff_max sampling_rate signal)

{- Funzione per il filtro passa basso:
   - il primo argomento è la frequenza di taglio;
   - il secondo è la frequenza di campionamento;
   - il terzo argomento è il segnale da filtrare. -}
low_pass_filter :: Double -> Double -> [Complex Double] -> [Complex Double]
low_pass_filter _ _ [] = []
low_pass_filter cutoff sr (x:xs) = x : low_pass_filter_helper cutoff sr 1 0 (length (x:xs)) xs

{- Funzione per applicare il filtro passa basso al segnale meno il primo elemento:
   - il primo argomento è la frequenza di taglio;
   - il secondo argomento è la frequenza di campionamento;
   - il terzo argomento è l'indice dell'elemento in valutazione;
   - il quarto argomento è un parametro di controllo per ottimizzare il calcolo,
     evitando di calcolare la frequenza associata all'elemento quando non necessario;
   - il quinto argomento è il numero degli elementi del segnale;
   - il sesto argomento è il segnale da filtrare. -}
low_pass_filter_helper :: Double -> Double -> Int -> Int -> Int -> [Complex Double] -> [Complex Double]
low_pass_filter_helper _ _ _ _ _ [] = []
low_pass_filter_helper cutoff sr index control size (x:xs)
  | control == 1 = 0 :+ imagPart x : low_pass_filter_helper cutoff sr (index + 1) control size xs
  | fq > cutoff = 0 :+ imagPart x : low_pass_filter_helper cutoff sr (index + 1) 1 size xs
  | otherwise = x : low_pass_filter_helper cutoff sr (index + 1) control size xs
    where
      fq = ((fromIntegral index / fromIntegral size) * sr)

{- Funzione per il filtro passa alto:
   - il primo argomento è la frequenza di taglio;
   - il secondo è la frequenza di campionamento;
   - il terzo argomento è il segnale da filtrare. -}
high_pass_filter :: Double -> Double -> [Complex Double] -> [Complex Double]
high_pass_filter _ _ [] = []
high_pass_filter cutoff sr (x:xs) = x : high_pass_filter_helper cutoff sr 1 (length (x:xs)) xs

{- Funzione per applicare il filtro passa alto al segnale meno il primo elemento:
   - il primo argomento è la frequenza di taglio;
   - il secondo argomento è la frequenza di campionamento;
   - il terzo argomento è l'indice dell'elemento in valutazione;
   - il quarto argomento è il numero degli elementi del segnale;
   - il quinto argomento è il segnale da filtrare. -}
high_pass_filter_helper :: Double -> Double -> Int -> Int -> [Complex Double] -> [Complex Double]
high_pass_filter_helper _ _ _ _ [] = []
high_pass_filter_helper cutoff sr index size (x:xs)
  | fq < cutoff = 0 :+ imagPart x : high_pass_filter_helper cutoff sr (index + 1) size xs
  | otherwise = (x:xs)
    where
      fq = ((fromIntegral index / fromIntegral size) * sr)

{- Funzione per il filtro passa banda:
   - il primo argomento è la frequenza di taglio inferiore;
   - il secondo argomento è la frequenza di taglio superiore;
   - il terzo elemento è la frequenza di campionamento;
   - il quarto elemento è il segnale da filtrare. -}
band_pass_filter :: Double -> Double -> Double -> [Complex Double] -> [Complex Double]
band_pass_filter cutoff_min cutoff_max sr signal = filtered_signal
  where
    low_passed_signal = low_pass_filter cutoff_max sr signal
    filtered_signal = high_pass_filter cutoff_min sr low_passed_signal

{- Funzione per applicare la trasformata discreta inversa di Fourier (IDFT):
   - il suo unico argomento è il segnale nel dominio della frequenza. -}
idft :: [Complex Double] -> [Complex Double]
idft [] = []
idft signal = idft_k_handler signal (length signal) 0

{- Funzione per la gestione dell'indice k per il calcolo della IDFT:
   - il primo argomento è il segnale nel dominio della frequenza;
   - il secondo argomento è la dimensione del segnale;
   - il terzo argomento è l'indice K, indica l'elemento a cui si sta facendo riferimento. -}
idft_k_handler :: [Complex Double] -> Int -> Int -> [Complex Double]
idft_k_handler [] _ _ = []
```

```
idft_k_handler signal size k
  | k == size = []
  | otherwise =
    (idft_k_sum signal size k 0) / fromIntegral size :
    (idft_k_handler signal size (k + 1))

{- Funzione per effettuare il calcolo del k-esimo elemento della trasformata inversa di Fourier:
   - il primo argomento è il segnale nel dominio della frequenza;
   - il secondo argomento è la dimensione del segnale;
   - il terzo argomento è l'indice K, indica l'elemento del segnale nel dominio della frequenza
     a cui si sta facendo riferimento;
   - il quarto argomento è l'indice N, indica l'elemento del segnale nel dominio del tempo
     a cui si sta facendo riferimento.  -}
idft_k_sum :: [Complex Double] -> Int -> Int -> Int -> Complex Double
idft_k_sum [] _ _ _ = 0.0
idft_k_sum (x:xs) size k n =
  x * (cos esp :+ sin esp) + idft_k_sum xs size k (n + 1)
  where
    esp = 2.0 * pi * fromIntegral n * fromIntegral k / fromIntegral size

{- Definizione delle funzioni ausiliarie -}

{- Funzione per acquisire un segnale fornito in input dall'utente,
   se il segnale non è valido viene richiesto un nuovo inserimento. -}
read_input_signal :: IO [Double]
read_input_signal = do
  input <- getLine
  case readMaybe input of
    Just xs -> return xs
    Nothing -> do
      putStrLn "\nInput non valido. Inserire una lista finita di numeri reali (es. [1, -2.5, 3.2]):"
      read_input_signal

{- Funzione per acquisire e validare un numero reale positivo fornito
   in input dall'utente, se il valore non è valido viene richiesto un nuovo inserimento. -}
read_double_value :: IO Double
read_double_value = do
  input <- getLine
  case readMaybe input of
    Just x | x > 0 -> return x
    Just _ -> do
      putStrLn "\nInput non valido. Inserire un numero reale positivo: "
      read_double_value
    Nothing -> do
      putStrLn "\nInput non valido. Inserire un numero reale positivo: "
      read_double_value

{- Funzione per acquisire e validare la scelta relativa al filtro da utilizzare,
   se l'inserimento non è valido, viene richiesto un nuovo inserimento. -}
read_filter_choice :: IO Int
read_filter_choice = do
  input <- getLine
  case input of
    "1" -> return 1
    "2" -> return 2
    "3" -> return 3
    _ -> do
      putStrLn "\nInput non valido. Inserire 1, 2 o 3."
      read_filter_choice

{- Funzione per convertire una lista di numeri reali in una lista di numeri complessi:
   - il suo unico argomento è la lista di numeri reali da convertire. -}
real_to_complex :: [Double] -> [Complex Double]
real_to_complex  = map (\x -> x :+ 0)

{- Funzione per convertire una lista di numeri complessi in una lista di numeri reali:
   - il suo unico argomento è la lista di numeri complessi da convertire. -}
complex_to_real :: [Complex Double] -> [Double]
complex_to_real xs = map realPart xs
```

4.2 Programma Prolog

File sorgente: DFT_Filter_IDFT.pl

```
/* Programma Prolog per calcolare la trasformata discreta di Fourier (DFT) di un segnale,
   applicare un filtro e, infine, calcolare la trasformata discreta di Fourier inversa (IDFT)
   del segnale filtrato. */

/* Main */
/* Il programma accetta in input una lista finita di numeri reali rappresentante un segnale
   nel dominio del tempo e la frequenza di campionamento del segnale stesso, converte la
   lista di numeri reali in una lista di numeri complessi, calcola e stampa la trasformata
   discreta di Fourier (DFT) del segnale.
   Successivamente richiede la tipologia di filtro (passa basso, passa alto o passa banda)
   da applicare al segnale nel dominio delle frequenze e la-e frequenza-e di taglio.
   Dopo aver stampato il segnale filtrato nel dominio della frequenza, calcola la trasformata
   discreta inversa di Fourier (IDFT) del segnale e infine stampa il segnale ottenuto nel dominio
   del tempo. */

main :-
  write('Progetto per il corso di Programmazione Logica e Funzionale'), nl,
  write('Sessione Autunnale 2022/2023'), nl,
  write('Studente: Susanna Peretti,  Matricola: 306186'), nl,
  write('Studente: Marzio Della Bosca, Matricola: 306034'), nl, nl, nl,

  write('Inserire il segnale da filtrare (es. [1, 2.3, -2]): '), nl,
  read_input_signal(RealInputSignal),
  real_to_complex(RealInputSignal, ComplexInputSignal), nl,

  write('Inserire la frequenza di campionamento del segnale: '), nl,
  read_double_value(SamplingRate), nl,

  write('Applicazione della trasformata discreta di Fourier (DFT).'), nl,
  dft(ComplexInputSignal, DftSignal),
  write('Segnale nel dominio della frequenza:'), nl,
  print_complex_list(0, DftSignal), nl, nl,

  write('Selezione del filtro da applicare al segnale:'), nl,
  write(' 1 - Filtro passa basso'), nl,
  write(' 2 - Filtro passa alto'), nl,
  write(' 3 - Filtro passa banda'), nl,
  write('Digitare 1, 2 o 3: '), nl,
  read_filter_choice(Filter), nl,
  filtering(Filter, SamplingRate, DftSignal, FilteredSignal), nl,

  write('Segnale filtrato nel dominio della frequenza:'), nl,
  print_complex_list(0, FilteredSignal), nl, nl,

  write('Applicazione della trasformata discreta di Fourier inversa (IDFT).'), nl,
  idft(FilteredSignal, IdftSignal),
  write('Segnale filtrato nel dominio del tempo:'), nl,
  complex_to_real(IdftSignal, TimeDomainFilteredSignal),
  write(TimeDomainFilteredSignal).


/* Definizione dei predicati */

/* Definizione dei predicati principali */

/* Predicato per applicare la trasformata discreta di Fourier (DFT):
   - il primo argomento è il segnale nel dominio del tempo;
   - il secondo argomento è il segnale convertito nel dominio delle frequenze. */
dft([], []).
dft([X | Xs], DFT) :-
  size([X | Xs], Size),
  dft_k_handler([X | Xs], Size, 0, DFT).

/* Predicato per la gestione dell'indice k per il calcolo della DFT:
   - il primo argomento è il segnale nel dominio del tempo;
   - il secondo argomento è la dimensione del segnale;
   - il terzo argomento è l'indice k, indica l'elemento a cui si sta facendo riferimento;
   - il quarto argomento è la DFT del segnale. */
dft_k_handler([], _, _, []).
```

```
dft_k_handler(InputSignal, Size, K, DFT) :-
  ((K == Size) ->
    DFT = []
  ;
    K1 is K + 1,
    dft_k_handler(InputSignal, Size, K1, DftTail),
    dft_k_sum(InputSignal, Size, K, 0, DftKElement),
    append([DftKElement], DftTail, DFT)).

/* Predicato per effettuare il calcolo del k-esimo elemento della DFT:
   - il primo argomento è il segnale nel dominio del tempo;
   - il secondo argomento è la dimensione del segnale;
   - il terzo argomento è l'indice k, indica l'elemento del segnale nel dominio della frequenza
     a cui si sta facendo riferimento;
   - il quarto argomento è l'indice n, indica l'elemento, del segnale nel dominio del tempo
     a cui si sta facendo riferimento.
   - il quinto argomento è il k-esimo elemento della DFT calcolato. */
dft_k_sum([], _, _, _, (0, 0)).
dft_k_sum([Head | Tail], Size, K, N, DftKElement) :-
  N1 is N + 1,
  Esp is (-2 * pi * N * K / Size),
  dft_k_sum(Tail, Size, K, N1, DftTail),
  complex_prod(Head, (cos(Esp), sin(Esp)), Prod),
  complex_sum(Prod, DftTail, DftKElement).

/* Predicato per acquisire e validare il-i valore-i della-e frequenza-e di taglio
   ed eseguire l'applicazione del filtro scelto:
   - il primo argomento rappresenta il filtro selezionato;
   - il secondo argomento è la frequenza di campionamento del segnale;
   - il terzo argomento è il segnale da filtrare;
   - il quarto argomento è il segnale filtrato. */
filtering(_, _, [], []).
filtering(1, SamplingRate, Signal, FilteredSignal) :-
    write('Inserire la frequenza di taglio '),
    write('per il filtro passa basso:'), nl,
    read_double_value(Cutoff),
    low_pass_filter(Cutoff, SamplingRate, Signal, FilteredSignal).
filtering(2, SamplingRate, Signal, FilteredSignal) :-
    write('Inserire la frequenza di taglio '),
    write('per il filtro passa alto:'), nl,
    read_double_value(Cutoff),
    high_pass_filter(Cutoff, SamplingRate, Signal, FilteredSignal).
filtering(3, SamplingRate, Signal, FilteredSignal) :-
    repeat,
    write('Inserire la frequenza di taglio inferiore: '), nl,
    read_double_value(CutoffMin), nl,
    write('Inserire la frequenza di taglio superiore: '), nl,
    read_double_value(CutoffMax),
    (CutoffMin >= CutoffMax ->
      write('Errore, la frequenza di taglio inferiore non deve '),
      write('essere maggiore o uguale della frequenza di taglio superiore!'),
      nl, nl, fail
    ;
      band_pass_filter(CutoffMin, CutoffMax, SamplingRate, Signal, FilteredSignal), !
    ).

/* Predicato per il filtro passa basso:
   - il primo argomento è la frequenza di taglio;
   - il secondo argomento è la frequenza di campionamento del segnale;
   - il terzo argomento è il segnale da filtrare;
   - il quarto argomento è il segnale filtrato. */
low_pass_filter(_, _, [], []).
low_pass_filter(Cutoff, SamplingRate, [Head | Rest], [Head | FilteredTail]) :-
  size([Head | Rest], Size),
  low_pass_helper(Cutoff, 1, 0, Size, SamplingRate, Rest, FilteredTail).

/* Predicato per applicare il filtro passa basso al segnale meno il primo elemento:
   - il primo argomento è la frequenza di taglio;
   - il secondo argomento è l'indice dell'elemento in valutazione;
   - il terzo argomento è un parametro di controllo per ottimizzare il calcolo,
     evitando di calcolare la frequenza associata all'elemento quando non necessario;
   - il quarto argomento è il numero di elementi del segnale;
   - il quinto argomento è la frequenza di campionamento del segnale;
```

```
   - il sesto argomento è il segnale da filtrare;
   - il settimo argomento è il segnale filtrato. */
low_pass_helper(_, _, _, _, _, [], []).
low_pass_helper(Cutoff, Index, 1, Size, SR, [(_, Img) | Tail], [FilteredHead | FilteredTail]) :-
  FilteredHead = (0, Img),
  low_pass_helper(Cutoff, Index + 1, 1, Size, SR, Tail, FilteredTail).
low_pass_helper(Cutoff, Index, 0, Size, SR, [Head | Tail], [FilteredHead | FilteredTail]) :-
  Head = (Real, Img),
  ((Index / Size * SR) > Cutoff ->
    FilteredHead = (0, Img),
    low_pass_helper(Cutoff, Index + 1, 1, Size, SR, Tail, FilteredTail)
  ;
    FilteredHead = (Real, Img),
    low_pass_helper(Cutoff, Index + 1, 0, Size, SR, Tail, FilteredTail)
  ).

/* Predicato per il filtro passa alto:
   - il primo argomento è la frequenza di taglio;
   - il secondo argomento è la frequenza di campionamento del segnale;
   - il terzo argomento è il segnale da filtrare;
   - il quarto argomento è il segnale filtrato. */
high_pass_filter(_, _, [], []).
high_pass_filter(Cutoff, SR, [Head | Rest], [Head | FilteredTail]) :-
  size([Head | Rest], Size),
  high_pass_helper(Cutoff, 1, Size, SR, Rest, FilteredTail).

/* Predicato per applicare il filtro passa alto al segnale meno il primo elemento:
   - il primo argomento è la frequenza di taglio;
   - il secondo argomento è l'indice dell'elemento in valutazione;
   - il terzo argomento è il numero degli elementi del segnale;
   - il quarto argomento è la frequenza di campionamento del segnale;
   - il quinto argomento è il segnale da filtrare;
   - il sesto argomento è il segnale filtrato. */
high_pass_helper(Cutoff, Index, Size, SR, [(Real, Img) | Tail], [FilteredHead | FilteredTail]) :-
  ((Index / Size * SR) < Cutoff ->
    FilteredHead = (0, Img),
    high_pass_helper(Cutoff, Index + 1, Size, SR, Tail, FilteredTail)
  ;
    FilteredHead = (Real, Img),
    FilteredTail = Tail
  ).

/* Predicato per il filtro passa banda:
   - il primo argomento è la frequenza di taglio inferiore;
   - il secondo argomento è è la frequenza di taglio superiore;
   - il terzo argomento è la frequenza di campionamento del segnale;
   - il quarto argomento è il segnale da filtrare;
   - il quinto argomento è il segnale filtrato. */
band_pass_filter(CutoffMin, CutoffMax, SR, Signal, FilteredSignal) :-
    low_pass_filter(CutoffMax, SR, Signal, LowPassFilterResult),
    high_pass_filter(CutoffMin, SR, LowPassFilterResult, FilteredSignal).

/* Predicato per applicare la trasformata discreta inversa di Fourier (IDFT):
   - il primo argomento è il segnale nel dominio della frequenza;
   - il secondo argomento è il segnale convertito nel dominio del tempo. */
idft([], []).
idft([X | Xs], IDFT) :-
  size([X | Xs], Size),
  idft_k_handler([X | Xs], Size, 0, IDFT).

/* Predicato per la gestione dell'indice k per il calcolo della IDFT:
   - il primo argomento è il segnale nel dominio della frequenza;
   - il secondo argomento è il numero di elementi segnale;
   - il terzo argomento è l'indice N, indica l'elemento a cui si sta facendo riferimento;
   - il quarto argomento è il segnale convertito nel dominio del tempo. */
idft_k_handler([], _, _, []).
idft_k_handler(InputSignal, Size, N, IDFT) :-
  ((N == Size) -> IDFT = []
  ;
    N1 is N + 1,
    idft_k_handler(InputSignal, Size, N1, IdftTail),
    idft_k_sum(InputSignal, Size, 0, N, IdftKElement),
    complex_div_real(IdftKElement, Size, Prod),
```

```
    append([Prod], IdftTail, IDFT)).

/* Predicato per effettuare il calcolo del k-esimo elemento della trasformata inversa di Fourier:
   - il primo argomento è il segnale nel dominio della frequenza;
   - il secondo argomento è il numero di elementi segnale;
   - il terzo argomento è l'indice K, indica l'elemento, del segnale nel dominio della frequenza
     a cui si sta facendo riferimento;
   - il quarto argomento è l'indice N, indica l'elemento, del segnale nel dominio del tempo
     a cui si sta facendo riferimento.
   - il quinto argomento è il k-esimo elemento della IDFT calcolato. */
idft_k_sum([], _, _, _, (0, 0)).
idft_k_sum([Head | Tail], Size, K, N, IdftKElement) :-
  K1 is K + 1,
  Esp is (2 * pi * K * N / Size),
  idft_k_sum(Tail, Size, K1, N, IdftTail),
  complex_prod(Head, (cos(Esp), sin(Esp)), Prod),
  complex_sum(Prod, IdftTail, IdftKElement).

/* Predicati ausiliari */

/* Predicato per acquisire un segnale fornito in input dall'utente,
   se il segnale non è valido viene richiesto un nuovo inserimento:
   - il suo unico argomento è il segnale validato. */
read_input_signal(List) :-
  repeat,
  catch(read(List), _, fail),
  (valid_real_list(List) ->
    !
  ;
    nl, write('Input non valido. Inserire una lista finita di numeri reali (es. [1, -2.5, 3.2]): '),
    nl, fail).

/* Predicato per validare una lista di numeri reali, verifica che ogni elemento presente sia
   un numero reale:
   - il suo unico argomento è la lista di numeri reali validata. */
valid_real_list([]).
valid_real_list([X | Rest]) :-
  number(X),
  valid_real_list(Rest).

/* Predicato per acquisire e validare un numero reale positivo fornito
   in input dall'utente, se il valore non è valido viene richiesto un nuovo inserimento:
   - il suo unico argomento è il valore della frequenza di taglio, rappresentato da un numero reale. */
read_double_value(Value) :-
    repeat,
    read(Value),
    (number(Value) ->
        (Value > 0 ->
            !
        ;
            nl, write('Input non valido. Inserire un numero reale positivo: '),
            nl, fail
        )
    ;
        nl, write('Input non valido. Inserire un numero reale positivo: '),
        nl, fail
    ).

/* Predicato per acquisire e validare la scelta relativa al filtro da parte dell'utente,
   se l'inserimento non è valido, viene richiesto un nuovo inserimento.
   - il suo unico argomento è il valore rappresentante la scelta del filtro. */
read_filter_choice(Filter) :-
  repeat,
  catch(read(Filter), _, fail),
  (member(Filter, [1, 2, 3])  ->
    !
  ;
    nl, write('Input non valido. Inserire 1, 2 o 3.'),
    nl, fail).

/* Predicato per convertire una lista di numeri reali in una lista di numeri complessi:
   - il primo argomento è la lista di numeri reali da convertire;
   - il secondo argomento è la lista di numeri complessi ottenuta. */
```

```
real_to_complex([], []).
real_to_complex([X | RealTail], [(X, 0) | ComplexTail]) :-
  real_to_complex(RealTail, ComplexTail).
real_to_complex([X], [(X, 0)]).

/* Predicato per convertire una lista di numeri complessi in una lista di numeri reali:
   - il primo argomento è la lista di numeri complessi da convertire;
   - il secondo argomento è la lista di numeri reali ottenuta. */
complex_to_real([], []).
complex_to_real([(Real, _) | Tail], [RealHead | RealTail]) :-
  RealHead is Real,
  complex_to_real(Tail, RealTail).

/* Predicato per stampare una lista di numeri complessi in modo leggibile:
   - il primo argomento è un valore di controllo per gestire la stampa;
   - il secondo argomento è la lista di numeri complessi da stampare. */
print_complex_list(_, []) :- write('[]').
print_complex_list(0, [(Real, Imag) | []]) :-
  format('[~w :+ ~w]', [Real, Imag]).
print_complex_list(0, [(Real, Imag) | ComplexTail]) :-
  format('[~w :+ ~w', [Real, Imag]),
  print_complex_list(1, ComplexTail).
print_complex_list(1, [(Real, Imag) | []]) :-
  format(', ~w :+ ~w]', [Real, Imag]).
print_complex_list(1, [(Real, Imag) | ComplexTail]) :-
  format(', ~w :+ ~w', [Real, Imag]),
  print_complex_list(1, ComplexTail).

/* Predicato per calcolare il prodotto tra due numeri complessi:
   - il primo argomento è il primo dei due numeri complessi;
   - il secondo argomento è il secondo dei due numeri complessi;
   - il terzo argomento è il prodotto dei due numeri complessi. */
complex_prod((Real1, Imag1), (Real2, Imag2), (ProdReal, ProdImag)) :-
  ProdReal is Real1 * Real2 - Imag1 * Imag2,
  ProdImag is Real1 * Imag2 + Imag1 * Real2.

/* Predicato per calcolare la divisione tra due numeri complessi:
   - il primo argomento è il dividendo;
   - il secondo argomento è il divisore;
   - il terzo argomento è quoziente. */
complex_div_real((Real, Imag), Divisor, (DivReal, DivImag)) :-
  DivReal is Real / Divisor,
  DivImag is Imag / Divisor.

/* Predicato per calcolare la somma tra due numeri complessi:
   - il primo argomento è il primo dei due numeri complessi;
   - il secondo argomento è il secondo dei due numeri complessi;
   - il terzo argomento è la somma dei due numeri complessi. */
complex_sum((Real1, Imag1), (Real2, Imag2), (SumReal, SumImag)) :-
  SumReal is Real1 + Real2,
  SumImag is Imag1 + Imag2.

/* Predicato per calcolare il numero di elementi in una lista:
   - il primo argomento è la lista di cui calcolare la dimensione;
   - il secondo argomento è la dimensione della lista summenzionata. */
size([], 0).
size([_ | Tail], Size) :-
  size(Tail, TailSize),
  Size is TailSize + 1.
```

5
Testing del Programma

Al fine di non appesantire troppo la lettura dei test è stato deciso di riportare l'intestazione solo nei test 1 dei rispettivi programmi Haskell e Prolog.

5.1 Testing del programma Haskell

Test Haskell 1

Verifica che data una lista vuota, selezionando il filtro passa basso si ritorni una lista vuota al termine del programma, evitando di chiedere la frequenza di taglio.

```
Progetto per il corso di Programmazione Logica e Funzionale.
Sessione Autunnale 2022/2023.
Studente: Marzio Della Bosca, Matricola: 306034.
Studente: Susanna Peretti,  Matricola: 306186.


Inserire il segnale da filtrare (es. [1, 2.3, -2]):
[]

Inserire la frequenza di campionamento del segnale:
1

Applicazione della trasformata discreta di Fourier (DFT).
Segnale nel dominio della frequenza:
[]

Selezione del filtro da applicare al segnale:
 1 - Filtro passa basso;
 2 - Filtro passa alto;
 3 - Filtro passa banda.
Digitare 1, 2 o 3:
1

Segnale filtrato nel dominio della frequenza:
[]

Applicazione della trasformata discreta di Fourier inversa (IDFT).
Segnale filtrato nel dominio del tempo:
[]
```

Test Haskell 2

Verifica del corretto funzionamento della trasformata discreta di Fourier (DFT) e trasformata discreta di Fourier inversa (IDFT), utilizzando il filtro passa basso con un valore di frequenza di taglio pari alla frequenza di campionamento in modo che il segnale non venga filtrato.

```
Inserire il segnale da filtrare (es. [1, 2.3, -2]):
[1.1,2,3.3]

Inserire la frequenza di campionamento del segnale:
1000
```

```
Applicazione della trasformata discreta di Fourier (DFT).
Segnale nel dominio della frequenza:
[6.4 :+ 0.0,(-1.5500000000000012) :+ 1.1258330249197692,(-1.549999999999998) :+ (-1.125833024919772)]

Selezione del filtro da applicare al segnale:
 1 - Filtro passa basso;
 2 - Filtro passa alto;
 3 - Filtro passa banda.
Digitare 1, 2 o 3:
1

Inserire la frequenza di taglio per il filtro passa basso:
1000

Segnale filtrato nel dominio della frequenza:
[6.4 :+ 0.0,(-1.5500000000000012) :+ 1.1258330249197692,(-1.549999999999998) :+ (-1.125833024919772)]

Applicazione della trasformata discreta di Fourier inversa (IDFT).
Segnale filtrato nel dominio del tempo:
[1.1000000000000005,2.0,3.3000000000000003]
```

Test Haskell 3

Verifica del corretto funzionamento della trasformata discreta di Fourier (DFT) e trasformata discreta di Fourier inversa (IDFT), utilizzando il filtro passa alto con un valore di frequenza di taglio pari a 1 in modo che il segnale non venga filtrato.

```
Inserire il segnale da filtrare (es. [1, 2.3, -2]):
[1.2,-3,4.5]

Inserire la frequenza di campionamento del segnale:
1000

Applicazione della trasformata discreta di Fourier (DFT).
Segnale nel dominio della frequenza:
[2.7 :+ 0.0,0.4499999999999975 :+ 6.495190528383288,0.45000000000000484 :+ (-6.495190528383292)]

Selezione del filtro da applicare al segnale:
 1 - Filtro passa basso;
 2 - Filtro passa alto;
 3 - Filtro passa banda.
Digitare 1, 2 o 3:
2

Inserire la frequenza di taglio per il filtro passa alto:
1

Segnale filtrato nel dominio della frequenza:
[2.7 :+ 0.0,0.4499999999999975 :+ 6.495190528383288,0.45000000000000484 :+ (-6.495190528383292)]

Applicazione della trasformata discreta di Fourier inversa (IDFT).
Segnale filtrato nel dominio del tempo:
[1.2000000000000008,-3.0,4.5]
```

Test Haskell 4

Verifica del corretto funzionamento del filtro passa basso.

```
Inserire il segnale da filtrare (es. [1, 2.3, -2]):
[1,-1,3,4]

Inserire la frequenza di campionamento del segnale:
1000

Applicazione della trasformata discreta di Fourier (DFT).
Segnale nel dominio della frequenza:
[7.0 :+ 0.0,(-2.000000000000001) :+ 5.0,1.0 :+ (-6.123233995736765e-16),(-1.9999999999999978) :+ (-5.000000000000001)]

Selezione del filtro da applicare al segnale:
 1 - Filtro passa basso;
```

```
 2 - Filtro passa alto;
 3 - Filtro passa banda.
Digitare 1, 2 o 3:
1

Inserire la frequenza di taglio per il filtro passa basso:
499

Segnale filtrato nel dominio della frequenza:
[7.0 :+ 0.0,(-2.000000000000001) :+ 5.0,0.0 :+ (-6.123233995736765e-16),0.0 :+ (-5.000000000000001)]

Applicazione della trasformata discreta di Fourier inversa (IDFT).
Segnale filtrato nel dominio del tempo:
[1.2499999999999998,-0.75,2.2500000000000004,4.25]
```

Test Haskell 5

Verifica del corretto funzionamento del filtro passa alto.

```
Inserire il segnale da filtrare (es. [1, 2.3, -2]):
[0,3.4,-2,1.1]

Inserire la frequenza di campionamento del segnale:
1000

Applicazione della trasformata discreta di Fourier (DFT).
Segnale nel dominio della frequenza:
[2.5 :+ 0.0,2.0 :+ (-2.3),(-6.5) :+ (-1.310372075087668e-15),1.9999999999999998 :+ 2.3000000000000007]

Selezione del filtro da applicare al segnale:
 1 - Filtro passa basso;
 2 - Filtro passa alto;
 3 - Filtro passa banda.
Digitare 1, 2 o 3:
2

Inserire la frequenza di taglio per il filtro passa alto:
501

Segnale filtrato nel dominio della frequenza:
[2.5 :+ 0.0,0.0 :+ (-2.3),0.0 :+ (-1.310372075087668e-15),1.9999999999999998 :+ 2.3000000000000007]

Applicazione della trasformata discreta di Fourier inversa (IDFT).
Segnale filtrato nel dominio del tempo:
[1.125,1.775,0.125,-0.5249999999999999]
```

Test Haskell 6

Verifica del corretto funzionamento del filtro passa banda.

```
Inserire il segnale da filtrare (es. [1, 2.3, -2]):
[1,-1,3,4]

Inserire la frequenza di campionamento del segnale:
1000

Applicazione della trasformata discreta di Fourier (DFT).
Segnale nel dominio della frequenza:
[7.0 :+ 0.0,(-2.000000000000001) :+ 5.0,1.0 :+ (-6.123233995736765e-16),(-1.9999999999999978) :+ (-5.000000000000001)]

Selezione del filtro da applicare al segnale:
 1 - Filtro passa basso;
 2 - Filtro passa alto;
 3 - Filtro passa banda.
Digitare 1, 2 o 3:
3

Inserire la frequenza di taglio inferiore:
450

Inserire la frequenza di taglio superiore:
550
```

```
Segnale filtrato nel dominio della frequenza:
[7.0 :+ 0.0,0.0 :+ 5.0,1.0 :+ (-6.123233995736765e-16),0.0 :+ (-5.000000000000001)]

Applicazione della trasformata discreta di Fourier inversa (IDFT).
Segnale filtrato nel dominio del tempo:
[2.0,-1.0,2.0000000000000004,4.0]
```

Test Haskell 7

Verifica del corretto funzionamento della validazione per l'inserimento della lista di numeri reali rappresentante il segnale.

```
Inserire il segnale da filtrare (es. [1, 2.3, -2]):
[a]

Input non valido. Inserire una lista finita di numeri reali (es. [1, -2.5, 3.2]):
[a,b,-2,1]

Input non valido. Inserire una lista finita di numeri reali (es. [1, -2.5, 3.2]):
[-2,2.3,!]

Input non valido. Inserire una lista finita di numeri reali (es. [1, -2.5, 3.2]):
[1,-1,3,4]

Inserire la frequenza di campionamento del segnale:
1000

Applicazione della trasformata discreta di Fourier (DFT).
Segnale nel dominio della frequenza:
[7.0 :+ 0.0,(-2.000000000000001) :+ 5.0,1.0 :+ (-6.123233995736765e-16),(-1.9999999999999978) :+ (-5.000000000000001)]

Selezione del filtro da applicare al segnale:
 1 - Filtro passa basso;
 2 - Filtro passa alto;
 3 - Filtro passa banda.
Digitare 1, 2 o 3:
1

Inserire la frequenza di taglio per il filtro passa basso:
499

Segnale filtrato nel dominio della frequenza:
[7.0 :+ 0.0,(-2.000000000000001) :+ 5.0,0.0 :+ (-6.123233995736765e-16),0.0 :+ (-5.000000000000001)]

Applicazione della trasformata discreta di Fourier inversa (IDFT).
Segnale filtrato nel dominio del tempo:
[1.2499999999999998,-0.75,2.2500000000000004,4.25]
```

Test Haskell 8

Verifica il corretto funzionamento della validazione per la scelta del filtro da applicare.

```
Inserire il segnale da filtrare (es. [1, 2.3, -2]):
[1,-1,3,4]

Inserire la frequenza di campionamento del segnale:
1000

Applicazione della trasformata discreta di Fourier (DFT).
Segnale nel dominio della frequenza:
[7.0 :+ 0.0,(-2.000000000000001) :+ 5.0,1.0 :+ (-6.123233995736765e-16),(-1.9999999999999978) :+ (-5.000000000000001)]

Selezione del filtro da applicare al segnale:
 1 - Filtro passa basso;
 2 - Filtro passa alto;
 3 - Filtro passa banda.
Digitare 1, 2 o 3:
a

Input non valido. Inserire 1, 2 o 3.
!
```

```
Input non valido. Inserire 1, 2 o 3.
-2

Input non valido. Inserire 1, 2 o 3.
1

Inserire la frequenza di taglio per il filtro passa basso:
499

Segnale filtrato nel dominio della frequenza:
[7.0 :+ 0.0,(-2.000000000000001) :+ 5.0,0.0 :+ (-6.123233995736765e-16),0.0 :+ (-5.000000000000001)]

Applicazione della trasformata discreta di Fourier inversa (IDFT).
Segnale filtrato nel dominio del tempo:
[1.2499999999999998,-0.75,2.2500000000000004,4.25]
```

Test Haskell 9

Verifica il corretto funzionamento della validazione per l'inserimento delle frequenze di taglio nel caso del filtro passa banda.

```
Inserire il segnale da filtrare (es. [1, 2.3, -2]):
[1,-1,3,4]

Inserire la frequenza di campionamento del segnale:
1000

Applicazione della trasformata discreta di Fourier (DFT).
Segnale nel dominio della frequenza:
[7.0 :+ 0.0,(-2.000000000000001) :+ 5.0,1.0 :+ (-6.123233995736765e-16),(-1.9999999999999978) :+ (-5.000000000000001)]

Selezione del filtro da applicare al segnale:
 1 - Filtro passa basso;
 2 - Filtro passa alto;
 3 - Filtro passa banda.
Digitare 1, 2 o 3:
3

Inserire la frequenza di taglio inferiore:
0

Input non valido. Inserire un numero reale positivo:
a

Input non valido. Inserire un numero reale positivo:
550

Inserire la frequenza di taglio superiore:
400
Errore, la frequenza di taglio inferiore non deve essere maggiore o uguale della frequenza di taglio superiore!

Inserire la frequenza di taglio inferiore:
550

Inserire la frequenza di taglio superiore:
550
Errore, la frequenza di taglio inferiore non deve essere maggiore o uguale della frequenza di taglio superiore!

Inserire la frequenza di taglio inferiore:
450

Inserire la frequenza di taglio superiore:
550

Segnale filtrato nel dominio della frequenza:
[7.0 :+ 0.0,0.0 :+ 5.0,1.0 :+ (-6.123233995736765e-16),0.0 :+ (-5.000000000000001)]

Applicazione della trasformata discreta di Fourier inversa (IDFT).
Segnale filtrato nel dominio del tempo:
[2.0,-1.0,2.0000000000000004,4.0]
```

Test Haskell 10

Verifica del corretto funzionamento della validazione per l'inserimento della frequenza di campionamento.

```
Inserire il segnale da filtrare (es. [1, 2.3, -2]):
[1,-1,3,4]

Inserire la frequenza di campionamento del segnale:
a

Input non valido. Inserire un numero reale positivo:
-2

Input non valido. Inserire un numero reale positivo:
!

Input non valido. Inserire un numero reale positivo:
abc

Input non valido. Inserire un numero reale positivo:
1000

Applicazione della trasformata discreta di Fourier (DFT).
Segnale nel dominio della frequenza:
[7.0 :+ 0.0,(-2.000000000000001) :+ 5.0,1.0 :+ (-6.123233995736765e-16),(-1.9999999999999978) :+ (-5.000000000000001)]

Selezione del filtro da applicare al segnale:
 1 - Filtro passa basso;
 2 - Filtro passa alto;
 3 - Filtro passa banda.
Digitare 1, 2 o 3:
1

Inserire la frequenza di taglio per il filtro passa basso:
499

Segnale filtrato nel dominio della frequenza:
[7.0 :+ 0.0,(-2.000000000000001) :+ 5.0,0.0 :+ (-6.123233995736765e-16),0.0 :+ (-5.000000000000001)]

Applicazione della trasformata discreta di Fourier inversa (IDFT).
Segnale filtrato nel dominio del tempo:
[1.2499999999999998,-0.75,2.2500000000000004,4.25]
```

5.2 Testing del programma Prolog

Test Prolog 1

Verifica che data una lista vuota, selezionando il filtro passa basso si ritorni una lista vuota al termine del programma, evitando di chiedere la frequenza di taglio.

```
Progetto per il corso di Programmazione Logica e Funzionale
Sessione Autunnale 2022/2023
Studente: Susanna Peretti,  Matricola: 306186
Studente: Marzio Della Bosca, Matricola: 306034


Inserire il segnale da filtrare (es. [1, 2.3, -2]):
[].

Inserire la frequenza di campionamento del segnale:
1.

Applicazione della trasformata discreta di Fourier (DFT).
Segnale nel dominio della frequenza:
[]

Selezione del filtro da applicare al segnale:
 1 - Filtro passa basso
 2 - Filtro passa alto
```

```
 3 - Filtro passa banda
Digitare 1, 2 o 3:
1.

Segnale filtrato nel dominio della frequenza:
[]

Applicazione della trasformata discreta di Fourier inversa (IDFT).
Segnale filtrato nel dominio del tempo:
[]
```

Test Prolog 2

Verifica del corretto funzionamento della trasformata discreta di Fourier (DFT) e trasformata discreta di Fourier inversa (IDFT), utilizzando il filtro passa basso con un valore di frequenza di taglio pari alla frequenza di campionamento in modo che il segnale non venga filtrato.

```
Inserire il segnale da filtrare (es. [1, 2.3, -2]):
[1.1,2,3.3].

Inserire la frequenza di campionamento del segnale:
1000.

Applicazione della trasformata discreta di Fourier (DFT).
Segnale nel dominio della frequenza:
[6.4000000000000004 :+ 0.0, -1.5500000000000012 :+ 1.1258330249197692, -1.549999999999998 :+ -1.1258330249197721]

Selezione del filtro da applicare al segnale:
 1 - Filtro passa basso
 2 - Filtro passa alto
 3 - Filtro passa banda
Digitare 1, 2 o 3:
1.

Inserire la frequenza di taglio per il filtro passa basso:
1000.

Segnale filtrato nel dominio della frequenza:
[6.4000000000000004 :+ 0.0, -1.5500000000000012 :+ 1.1258330249197692, -1.549999999999998 :+ -1.1258330249197721]

Applicazione della trasformata discreta di Fourier inversa (IDFT).
Segnale filtrato nel dominio del tempo:
[1.1000000000000003,2.0,3.3000000000000003]
```

Test Prolog 3

Verifica del corretto funzionamento della trasformata discreta di Fourier (DFT) e trasformata discreta di Fourier inversa (IDFT), utilizzando il filtro passa alto con un valore di frequenza di taglio pari a 1 in modo che il segnale non venga filtrato.

```
Inserire il segnale da filtrare (es. [1, 2.3, -2]):
[1.2,-3,4.5].

Inserire la frequenza di campionamento del segnale:
1000.

Applicazione della trasformata discreta di Fourier (DFT).
Segnale nel dominio della frequenza:
[2.7000000000000002 :+ 0.0, 0.44999999999999751 :+ 6.4951905283832883, 0.45000000000000484 :+ -6.4951905283832918]

Selezione del filtro da applicare al segnale:
 1 - Filtro passa basso
 2 - Filtro passa alto
 3 - Filtro passa banda
Digitare 1, 2 o 3:
2.

Inserire la frequenza di taglio per il filtro passa alto:
```

```
1.

Segnale filtrato nel dominio della frequenza:
[2.7000000000000002 :+ 0.0, 0.44999999999999751 :+ 6.4951905283832883, 0.45000000000000484 :+ -6.4951905283832918]

Applicazione della trasformata discreta di Fourier inversa (IDFT).
Segnale filtrato nel dominio del tempo:
[1.2000000000000008,-3.0,4.5]
```

Test Prolog 4

Verifica del corretto funzionamento per il filtro passa basso.

```
Inserire il segnale da filtrare (es. [1, 2.3, -2]):
[1,-1,3,4].

Inserire la frequenza di campionamento del segnale:
1000.

Applicazione della trasformata discreta di Fourier (DFT).
Segnale nel dominio delle frequenze:
[7.0 :+ 0.0, -2.0000000000000009 :+ 5.0, 1.0 :+ -6.1232339957367653e-16, -1.9999999999999978 :+ -5.0000000000000009]

Selezione del filtro da applicare al segnale:
 1 - Filtro passa basso
 2 - Filtro passa alto
 3 - Filtro passa banda
Digitare 1, 2 o 3:
1.

Inserire la frequenza di taglio per il filtro passa basso:
499.

Segnale filtrato nel dominio della frequenza:
[7.0 :+ 0.0, -2.0000000000000009 :+ 5.0, 0 :+ -6.1232339957367653e-16, 0 :+ -5.0000000000000009]

Applicazione della trasformata discreta di Fourier inversa (IDFT).
Segnale filtrato nel dominio del tempo:
[1.2499999999999998,-0.75,2.2500000000000004,4.25]
```

Test Prolog 5

Verifica del corretto funzionamento per il filtro passa alto.

```
Inserire il segnale da filtrare (es. [1, 2.3, -2]):
[0,3.4,-2,1.1].

Inserire la frequenza di campionamento del segnale:
1000.

Applicazione della trasformata discreta di Fourier (DFT).
Segnale nel dominio della frequenza:
[2.5 :+ 0.0, 2.0 :+ -2.2999999999999998, -6.5 :+ -1.3103720750876679e-15, 1.9999999999999998 :+ 2.3000000000000007]

Selezione del filtro da applicare al segnale:
 1 - Filtro passa basso
 2 - Filtro passa alto
 3 - Filtro passa banda
Digitare 1, 2 o 3:
2.

Inserire la frequenza di taglio per il filtro passa alto:
501.

Segnale filtrato nel dominio della frequenza:
[2.5 :+ 0.0, 0 :+ -2.2999999999999998, 0 :+ -1.3103720750876679e-15, 1.9999999999999998 :+ 2.3000000000000007]

Applicazione della trasformata discreta di Fourier inversa (IDFT).
Segnale filtrato nel dominio del tempo:
[1.125,1.7749999999999999,0.125,-0.52499999999999991]
```

Test Prolog 6

Verifica del corretto funzionamento per il filtro passa banda.

```
Inserire il segnale da filtrare (es. [1, 2.3, -2]):
[1,-1,3,4].

Inserire la frequenza di campionamento del segnale:
1000.

Applicazione della trasformata discreta di Fourier (DFT).
Segnale nel dominio della frequenza:
[7.0 :+ 0.0, -2.0000000000000009 :+ 5.0, 1.0 :+ -6.1232339957367653e-16, -1.9999999999999978 :+ -5.0000000000000009]

Selezione del filtro da applicare al segnale:
 1 - Filtro passa basso
 2 - Filtro passa alto
 3 - Filtro passa banda
Digitare 1, 2 o 3:
3.

Inserire la frequenza di taglio inferiore:
450.

Inserire la frequenza di taglio superiore:
550.

Segnale filtrato nel dominio della frequenza:
[7.0 :+ 0.0, 0 :+ 5.0, 1.0 :+ -6.1232339957367653e-16, 0 :+ -5.0000000000000009]

Applicazione della trasformata discreta di Fourier inversa (IDFT).
Segnale filtrato nel dominio del tempo:
[2.0,-1.0,2.0000000000000004,4.0]
```

Test Prolog 7

Verifica del corretto funzionamento della validazione per l'inserimento della lista di numeri reali rappresentante il segnale.

```
Inserire il segnale da filtrare (es. [1, 2.3, -2]):
[a].

Input non valido. Inserire una lista finita di numeri reali (es. [1, -2.5, 3.2]):
[a,b,-2,1].

Input non valido. Inserire una lista finita di numeri reali (es. [1, -2.5, 3.2]):
[-2,2.3,!].

Input non valido. Inserire una lista finita di numeri reali (es. [1, -2.5, 3.2]):
[1,-1,3,4].

Inserire la frequenza di campionamento del segnale:
1000.

Applicazione della trasformata discreta di Fourier (DFT).
Segnale nel dominio della frequenza:
[7.0 :+ 0.0, -2.0000000000000009 :+ 5.0, 1.0 :+ -6.1232339957367653e-16, -1.9999999999999978 :+ -5.0000000000000009]

Selezione del filtro da applicare al segnale:
 1 - Filtro passa basso
 2 - Filtro passa alto
 3 - Filtro passa banda
Digitare 1, 2 o 3:
1.

Inserire la frequenza di taglio per il filtro passa basso:
499.

Segnale filtrato nel dominio della frequenza:
[7.0 :+ 0.0, -2.0000000000000009 :+ 5.0, 0 :+ -6.1232339957367653e-16, 0 :+ -5.0000000000000009]

Applicazione della trasformata discreta di Fourier inversa (IDFT).
Segnale filtrato nel dominio del tempo:
```

```
[1.2499999999999998,-0.75,2.2500000000000004,4.25]
```

Test Prolog 8

Verifica del corretto funzionamento della validazione per la scelta del filtro da applicare.

```
Inserire il segnale da filtrare (es. [1, 2.3, -2]):
[1,-1,3,4].

Inserire la frequenza di campionamento del segnale:
1000.

Applicazione della trasformata discreta di Fourier (DFT).
Segnale nel dominio della frequenza:
[7.0 :+ 0.0, -2.0000000000000009 :+ 5.0, 1.0 :+ -6.1232339957367653e-16, -1.9999999999999978 :+ -5.0000000000000009]

Selezione del filtro da applicare al segnale:
 1 - Filtro passa basso
 2 - Filtro passa alto
 3 - Filtro passa banda
Digitare 1, 2 o 3:
a.

Input non valido. Inserire 1, 2 o 3.
!.

Input non valido. Inserire 1, 2 o 3.
-2.

Input non valido. Inserire 1, 2 o 3.
1.

Inserire la frequenza di taglio per il filtro passa basso:
499.

Segnale filtrato nel dominio della frequenza:
[7.0 :+ 0.0, -2.0000000000000009 :+ 5.0, 0 :+ -6.1232339957367653e-16, 0 :+ -5.0000000000000009]

Applicazione della trasformata discreta di Fourier inversa (IDFT).
Segnale filtrato nel dominio del tempo:
[1.2499999999999998,-0.75,2.2500000000000004,4.25]
```

Test Prolog 9

Verifica del corretto funzionamento della validazione per l'inserimento delle frequenze di taglio nel caso del filtro passa banda.

```
Inserire il segnale da filtrare (es. [1, 2.3, -2]):
[1,-1,3,4].

Inserire la frequenza di campionamento del segnale:
1000.

Applicazione della trasformata discreta di Fourier (DFT).
Segnale nel dominio della frequenza:
[7.0 :+ 0.0, -2.0000000000000009 :+ 5.0, 1.0 :+ -6.1232339957367653e-16, -1.9999999999999978 :+ -5.0000000000000009]

Selezione del filtro da applicare al segnale:
 1 - Filtro passa basso
 2 - Filtro passa alto
 3 - Filtro passa banda
Digitare 1, 2 o 3:
3.

Inserire la frequenza di taglio inferiore:
0.

Input non valido. Inserire un numero reale positivo:
a.

Input non valido. Inserire un numero reale positivo:
550.
```

```
Inserire la frequenza di taglio superiore:
400.
Errore, la frequenza di taglio inferiore non deve essere maggiore o uguale della frequenza di taglio superiore!

Inserire la frequenza di taglio inferiore:
550.

Inserire la frequenza di taglio superiore:
550.
Errore, la frequenza di taglio inferiore non deve essere maggiore o uguale della frequenza di taglio superiore!

Inserire la frequenza di taglio inferiore:
450.

Inserire la frequenza di taglio superiore:
550.

Segnale filtrato nel dominio della frequenza:
[7.0 :+ 0.0, 0 :+ 5.0, 1.0 :+ -6.1232339957367653e-16, 0 :+ -5.0000000000000009]

Applicazione della trasformata discreta di Fourier inversa (IDFT).
Segnale filtrato nel dominio del tempo:
[2.0,-1.0,2.0000000000000004,4.0]
```

Test Prolog 10

Verifica del corretto funzionamento della validazione per l'inserimento della frequenza di campionamento.

```
Inserire il segnale da filtrare (es. [1, 2.3, -2]):
[1,-1,3,4].

Inserire la frequenza di campionamento del segnale:
a.

Input non valido. Inserire un numero reale positivo:
-2.

Input non valido. Inserire un numero reale positivo:
!.

Input non valido. Inserire un numero reale positivo:
abc.

Input non valido. Inserire un numero reale positivo:
1000.

Applicazione della trasformata discreta di Fourier (DFT).
Segnale nel dominio della frequenza:
[7.0 :+ 0.0, -2.0000000000000009 :+ 5.0, 1.0 :+ -6.1232339957367653e-16, -1.9999999999999978 :+ -5.0000000000000009]

Selezione del filtro da applicare al segnale:
 1 - Filtro passa basso
 2 - Filtro passa alto
 3 - Filtro passa banda
Digitare 1, 2 o 3:
1.

Inserire la frequenza di taglio per il filtro passa basso:
499.

Segnale filtrato nel dominio della frequenza:
[7.0 :+ 0.0, -2.0000000000000009 :+ 5.0, 0 :+ -6.1232339957367653e-16, 0 :+ -5.0000000000000009]

Applicazione della trasformata discreta di Fourier inversa (IDFT).
Segnale filtrato nel dominio del tempo:
[1.2499999999999998,-0.75,2.2500000000000004,4.25]
```