Programmazione I e Laboratorio

Prova di laboratorio, Appello VI

(Tempo a disposizione - 2h)

Dato lo scheletro di codice qui riportato, lo si completi aggiungendo l'implementazione delle funzioni richieste.

bozza.c

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

//Functions to be implemented:
float *readArray(int *len);
float **computeMatrixZero(float a1[], float a2[], int len1, int
    len2);
float **transposeMatrix(float **A, int nrow, int ncol);

int main() {
  float *V1, *V2, **A, **A1;
  int len1, len2, i, j;

  //Reads and prints the two arrays:
  V1 = readArray(&len1);
  printf("Array:\n");
  for (i = 0; i<len1; i++) {
    printf("%.1f ", V1[i]);
  }
  printf("\n");
  V2 = readArray(&len2);
  printf("Array:\n");
  for (i = 0; i<len2; i++) {
    printf("%.1f ", V2[i]);
  }
  printf("\n");

  //Multiply the two arrays to obtain a matrix:
  A = computeMatrixZero(V1, V2, len1, len2);

  //Print the concatenated list:
  printf("Product Matrix:\n");
  for (i = 0; i<len1; i++) {
```

```
    float *row = A[i];
    for (j = 0; j<len2 - 1; j++)
      printf("%.1f ", row[j]);
    printf("%.1f\n", row[len2 - 1]);
  }

  //Deletes element out of order in the list
  A1 = transposeMatrix(A, len1, len2);

  //Print the cleaned list:
  printf("Transposed Matrix:\n");
  for (i = 0; i<len2; i++) {
    float *row = A1[i];
    for (j = 0; j<len1 - 1; j++)
      printf("%.1f ", row[j]);
    printf("%.1f\n", row[len1 - 1]);
  }

  return 0;
}
```

Le funzioni da implementare devono rispettare le seguenti specifiche:

- `readArray`: Legge dallo standard input una sequenza di numeri in virgola mobile e termina l'acquisizione quando viene letto un numero con parte decimale (i.e. a destra della virgola) diversa da zero. I numeri devono essere memorizzati, nell'ordine di acquisizione, in un array. La funzione restituisce il puntatore al primo elemento dell'array e scrive, nell'indirizzo puntato da `len`, la dimensione dell'array letto. Si puo' assumere che il vettore possa prevedere un massimo di 20 elementi.

- `computeMatrixZero`: dati due array $\mathbf{x}$ e $\mathbf{y}$ di dimensione, rispettivamente, m ed n, la funzione calcola la *matrice* ottenuta dal prodotto (outer product) tra il primo vettore ed il trasposto del secondo. Il risultato è un matrice A di $m \times n$ float ottenuta come segue:

$$A = \mathbf{x}\mathbf{y}^T = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \cdot \begin{bmatrix} y_1 & y_2 & \dots & y_n \end{bmatrix} = \begin{bmatrix} x_1y_1 & x_1y_2 & \dots & x_1y_n \\ x_2y_1 & x_2y_2 & \dots & x_2y_n \\ \dots & \dots & \dots & \dots \\ x_my_1 & x_my_2 & \dots & x_my_n \end{bmatrix}$$

  Data la matrice A così calcolata, la funzione identifica l'elemento massimo al suo interno (considerando il segno) e lo imposta a 0. Se due elementi della matrice hanno entrambi valore pari al massimo, va azzerato solo il primo elemento incontrato, ovvero quello con coppia di indici i, j più piccola secondo l'ordinamento lessicografico. La funzione prende come parametro i puntatori ai due vettori $\mathbf{x}$ e $\mathbf{y}$ e le loro dimensioni m ed n e restituisce un puntatore ad A.

- `transposeMatrix`: prende come parametro di input la matrice di float A con numero di righe m e numero di colonne n ed esegue la sua trasposta, ovvero calcola la matrice $A1 = A^T$ ottenuta a partire da A scambiandone le righe con le colonne. Il risultato è un matrice $A1$ di $n \times m$ float ottenuta come segue:

$$A1 = A^T = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}^T = \begin{bmatrix} a_{11} & a_{21} & \dots & a_{m1} \\ a_{12} & a_{22} & \dots & a_{m2} \\ \dots & \dots & \dots & \dots \\ a_{1n} & a_{2n} & \dots & a_{mn} \end{bmatrix}$$

  La funzione prende come parametro il puntatore alla matrice originale A ed il suo numero di righe m e colonne n; la funzione restituisce un puntatore ad $A1$.

Esempio

Input

```
1.0
2.0
-1.0
2.0
1.1
5.0
1.0
1.2
```

Output

```
Array:
1.0 2.0 -1.0 2.0
Array:
5.0 1.0
Product Matrix:
5.0 1.0
0.0 2.0
-5.0 -1.0
10.0 2.0
Transposed Matrix:
5.0 0.0 -5.0 10.0
1.0 2.0 -1.0 2.0
```