Esercizio 2 del 13/4/2016

Questo esercizio richiede di fare un programma distribuito su due file e quindi useremo la tecnica del file header (.h) vista a lezione. In realtà il programma consisterà nell'assemblaggio (con qualche modifica) delle soluzioni di 2 esercizi precedenti: l'esercizio_1_13_4_2016 e l'esercizio_3_6_4_2016. Quest'ultimo esercizio richiede una funzione F che , per un array dato A, un array M[100][2] che ha nella prima colonna i valori distinti che appaiono in A e nella seconda colonna il n. di occorrenze in A di ciascuno dei valori. Osserviamo che questo array M può venire usato per stabilire se un array A è contenuto, tenendo conto della numerosità dei valori, in un altro array B. Per fare questo test di inclusione infatti, basta costruire un array M per A e un simile array M' per B e poi confrontare M ed M'. Cosa significhi precisamente "confrontare M ed M' " dovrete formalizzarlo voi in una funzione che chiameremo G.

In questo esercizio avremo un array int X[400], che vedremo come int Y[lim1][lim2][lim3] e vogliamo costruire un array bool Z[lim2][lim3] tale che Z[i][j]=true sse l'H-fetta i è contenuta nella V-fetta j tenendo conto della numerosità degli elementi. Naturalmente vanno considerati solo gli elementi definiti delle fette. Per farlo useremo l'approccio delineato prima.

Quindi in un file, diciamo file1.cpp, avremo il main (dato) e una nuova funzione F (simile a quella dell'esercizio_3_6_4_2016) capace di costruire un array M corrispondente ad una H-fetta o a una V-fetta di Y. Per farlo questa nuova F deve usare le funzioni FH e FV dell'esercizio_1_13_4_2016 esattamente come previsto in quell'esercizio. Le 2 funzioni FH e FV dovranno risiedere su un altro file, diciamo file2.cpp, mentre le loro intestazioni e la definizione del tipo struttura E saranno su un file2 .h che dovrà venire incluso da file1.cpp e anche da file3.cpp.

La funzione G può risiedere su un terzo file file3.cpp che avrà anch'esso un file3.h che andrà incluso in file1.cpp.

Correttezza: specificare PRE e POST di G e dimostrare che G è corretta rispetto ad esse. Visto che G conterrà un ciclo, sarà necessario specificare un invariante ed una post-condizione di quel ciclo e poi dovrete dimostrare la correttezza del ciclo.

Purtroppo il moodle non prevede la consegna di file separati. Quindi per il test automatico dovrete mettere tutte le funzioni su un unico file. Nel seguito sono specificati i valori di alcuni test in modo che possiate sperimentare sul vostro PC il funzionamento della divisione del programma in più file.

case=A
input=
3 5 4 57
0 1 2 0
4 1 1 0
3 4 2 1
2 3 4 0
1 2 3 4

0 1 2 3
1 0 1 2
3 4 0 1
2 3 4 0
1 2 3 4

0 1 2 3
4 0 1 2
3 4 0 1
2 3 4 2
0
output=
1 0 0 1
0 0 0 0
0 1 0 0
0 0 0 0
1 1 1 1
end

case=B
input=
3 5 4 57
0 5 2 0
4 1 1 0
3 4 2 1
2 3 4 0
1 2 3 4

0 1 2 3
1 0 1 2
3 4 0 1
2 3 4 0
1 2 3 4

0 1 2 3
4 0 1 2
3 4 0 1
2 3 4 2
5
output=
0 0 0 0
0 0 0 0
0 1 0 0
0 0 0 0
1 1 0 0
end