CSC2040 Assessed Practical 5 Feedback

Maire Bowler <M.Bowler@qub.ac.uk>

Fri 27/04/2018 13:14

To: Dewei Liu <dliu08@qub.ac.uk>;

CSC2040 Data Structures and Algorithms and Programming Languages

Assessed Practical 5 Feedback Sheet

Student Name: Dewei Liu

Student Number: 40216004

Part 1 High-Dimensional Arrays and File I/O	
1. Two-D arrays (1) dynamic 2-d double array created of size N×M	correct
(2) array populated with random numbers within the range from 0 to maxV inclusive – Should be array[n][m] = rand() % (maxV + 1); (Note: rand()%maxV is only partially correct as 0<=rand()%maxV<=maxV-1 not including maxV)	correct
(3) average value of all the elements calculated	correct
(4) array deleted	correct
(5) average value found and returned	correct
Demonstrate that your function works by calling it with appropriate N, M and maxV values.	correct
2. Three-D arrays (1) dynamic 3-d char array created of size N×M×L	correct
(2) array initialised with random characters each with a random ASCII code value 'a'+rand()%26;	correct
(3) number of occurrences of the given character c counted	correct
(4) array deleted;	correct
(5) count returned	correct
A check of the validity of some parameters done	not implemented

3. File I/O area calculated	correct
File created in *text* mode and area saved to it	correct
File created in binary mode and area saved to it	correct
Code added to each function to open the file, read the information and print out information using cout	correct

Part 2 Hash Function	
Hash function uses a loop or a number of loops to access the elements of each of the parameters OR a key	correct
Hash_index calculated	issue within get_index and raising to a power which is always zero
The function is called correctly from a main function	correct
Check with the following values: street: "Butterfield"; city = "Dublin"; postcode = "BT266PP"; length: 3; tableSize: 11. Result: hash_index = 3	result slightly off

Part 3 Graphs	
Graph.txt 7 u 0 1 3.0 0 2 6.0 0 3 2.0 1 2 1.0 1 4 6.0 2 3 3.0 2 4 4.0 3 4 8.0 3 5 7.0 4 5 3.0 4 6 2.0 5 6 2.0	correct
The following tasks are performed by the function testGraph **Task 1:** A Graph object is created (may be dynamic)	correct
Task 2: Code provided to search the graph to locate the edge with the lowest weight > 0.0: as follows Double loop provided for edges from sources to destinations	undirected version possible source of mix up with results
If statement provided to check if the graph is directed or not	correct
If statement provided to get the lowest weight	correct
Task 3: Code is written to output the found lowest-weight edge in the format: “Source-vertex Destination-vertex Weight” For the graph from Fig 1 the output is 1 2 1	result returned was 2 1 1

TOTAL : 85