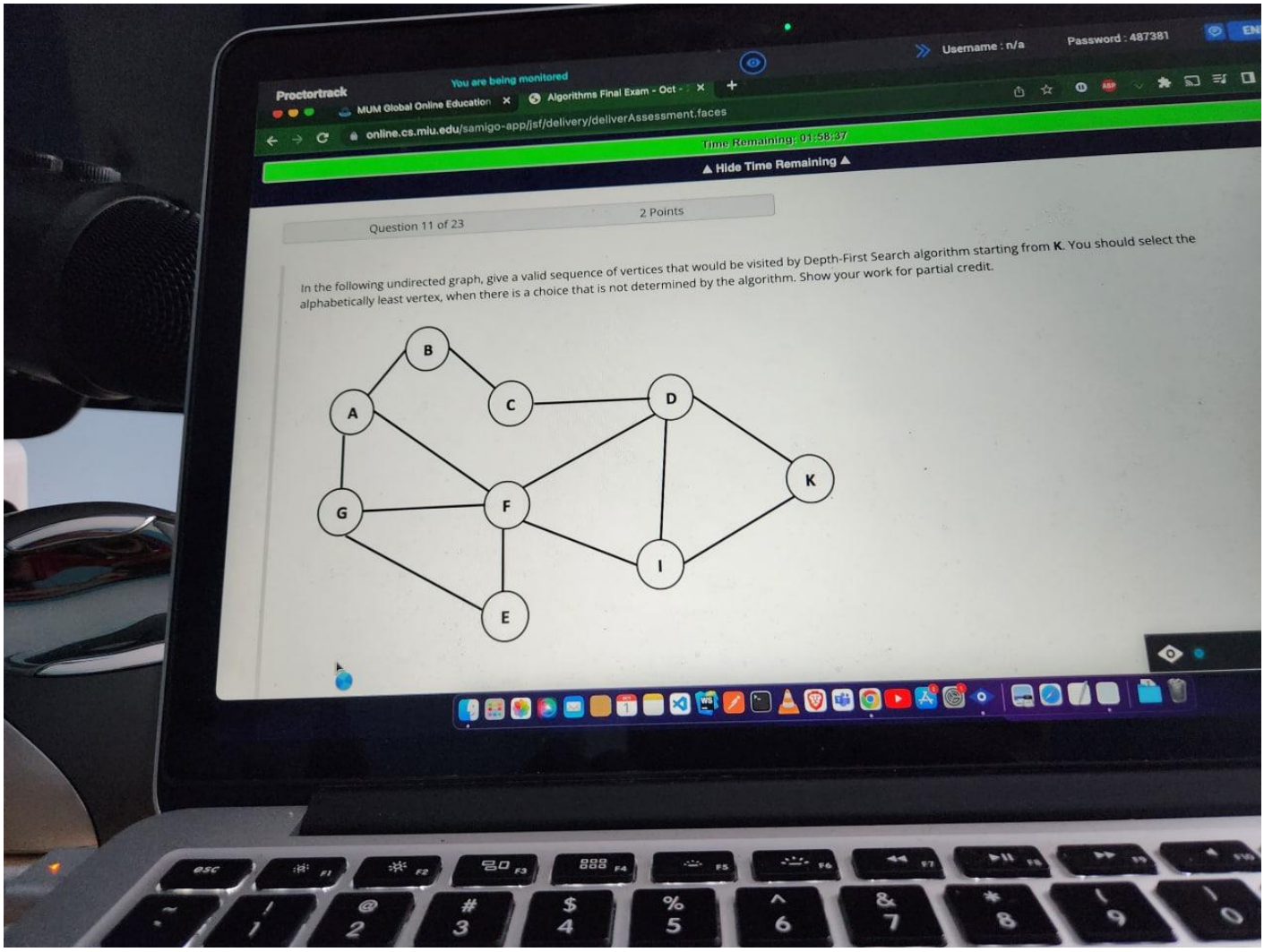
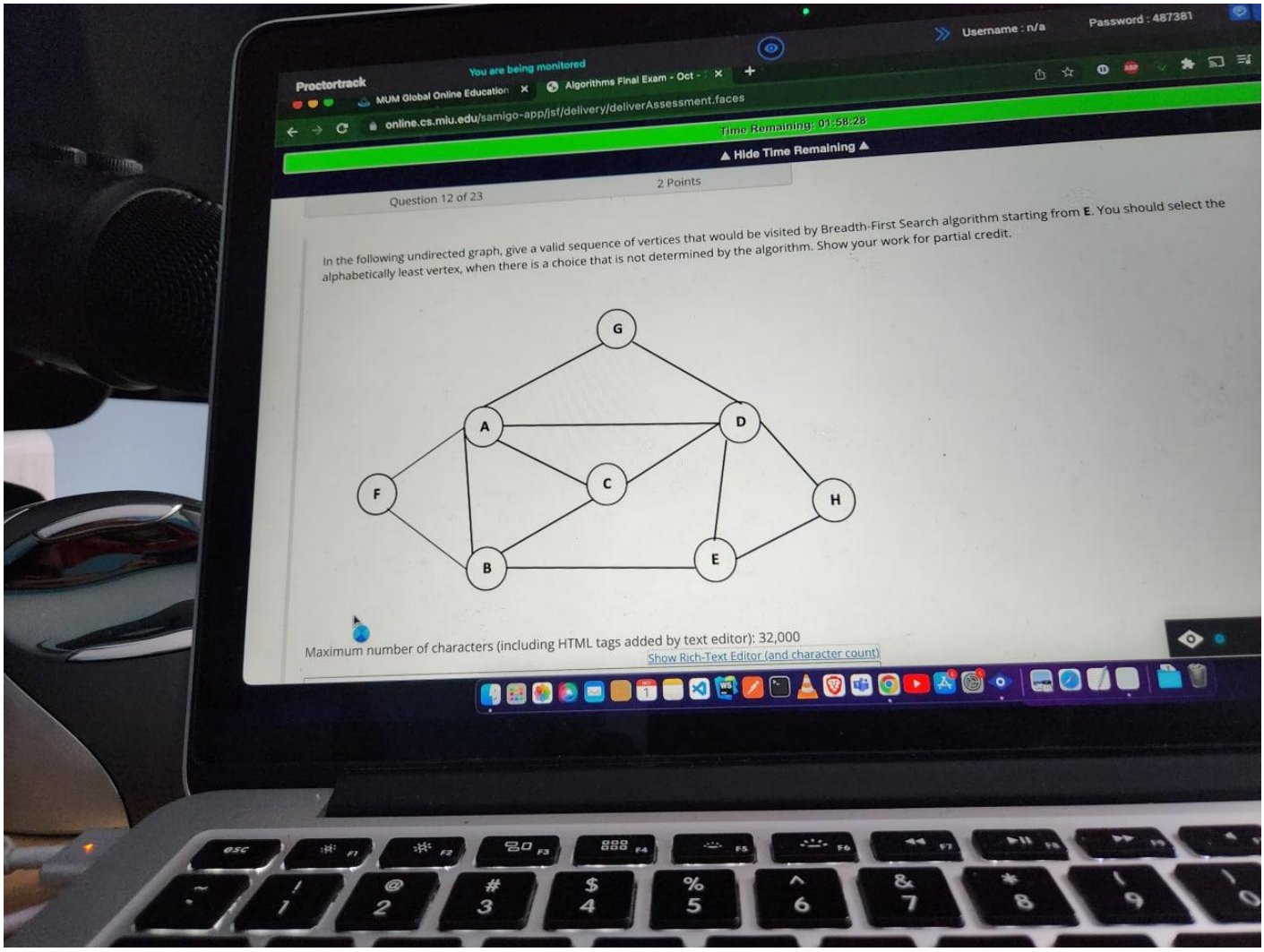
1. False
2. TRUE
3. True
4. TRUE
5. False
6. TRUE
7. FALSE
8. True
9. False
10. False



Question 11

K->D->C->B->A->F->E->G->I —-- Confirmed



Question 12

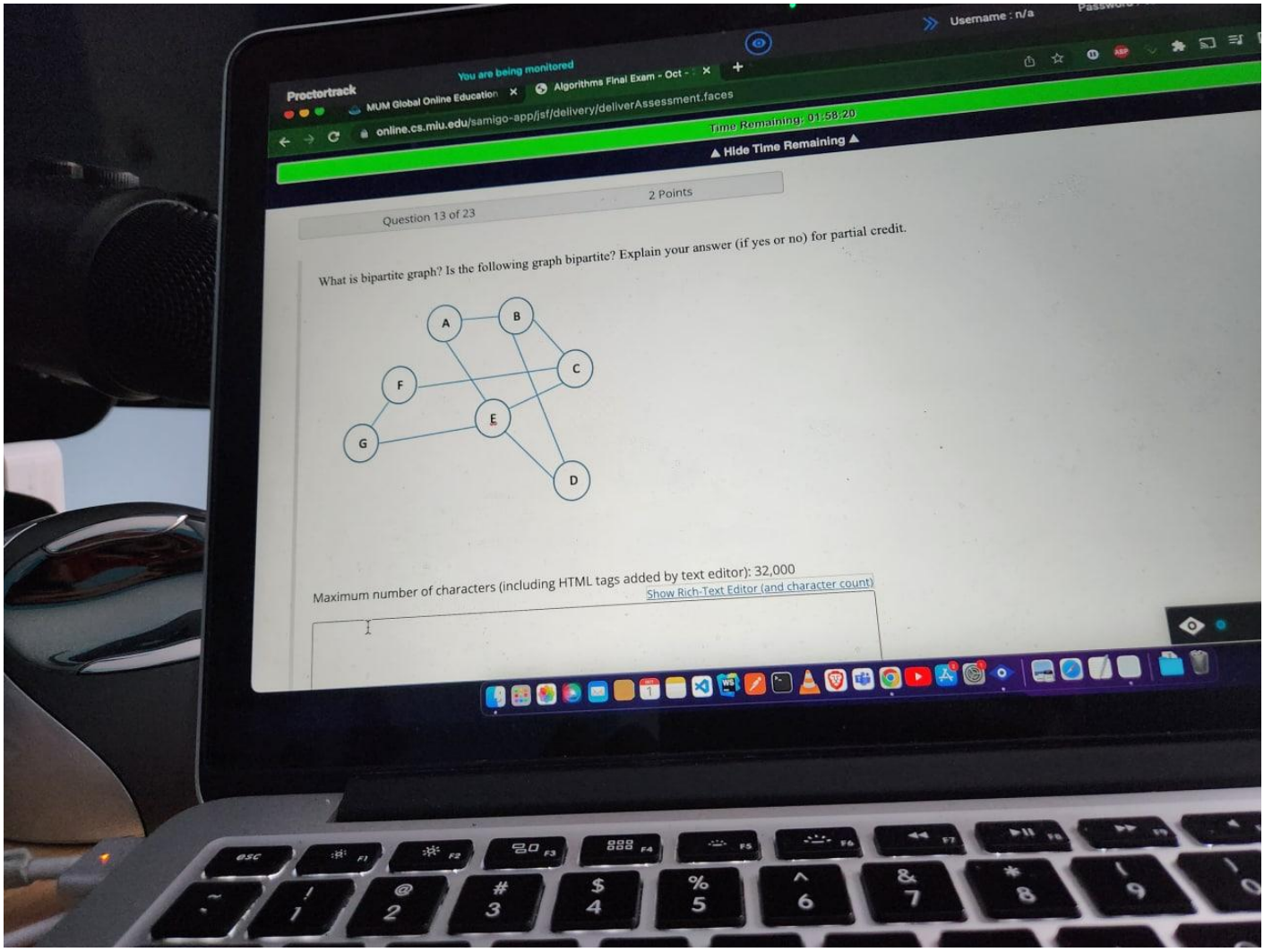
Answer

E -> B -> D -> H -> A -> C -> F -> G

First: E->B, E->D, E->H

Second: B->A, B->C, B->F

Third / Finally: D - > G

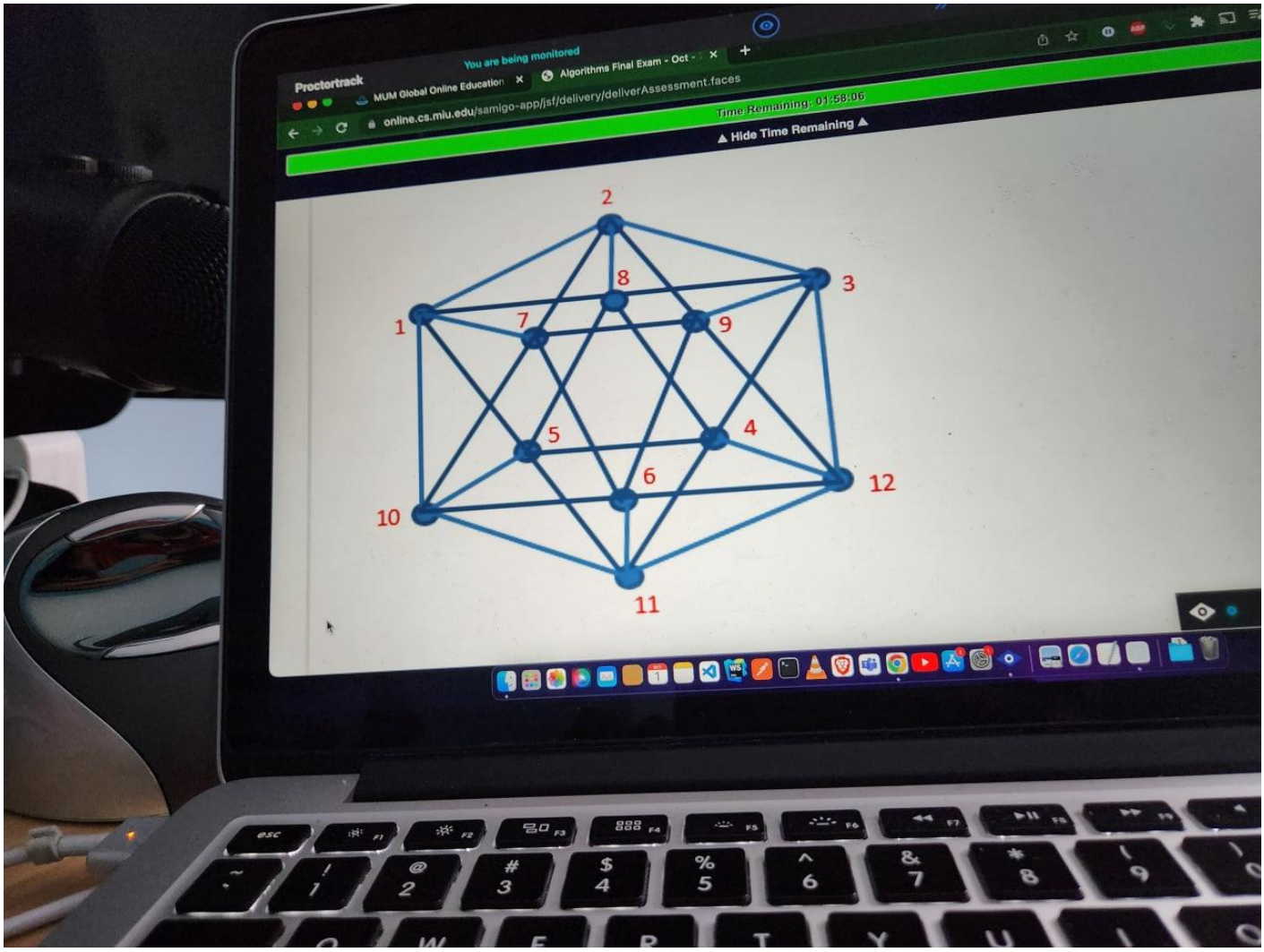


Question 13

Answer

A bipartite graph, also called a bigraph, is a set of graph vertices decomposed into two disjoint sets such that no two graph vertices within the same set are adjacent.

Yes this is a bipartite graph because we can put [FEB] in one set and [GDCA] in another set

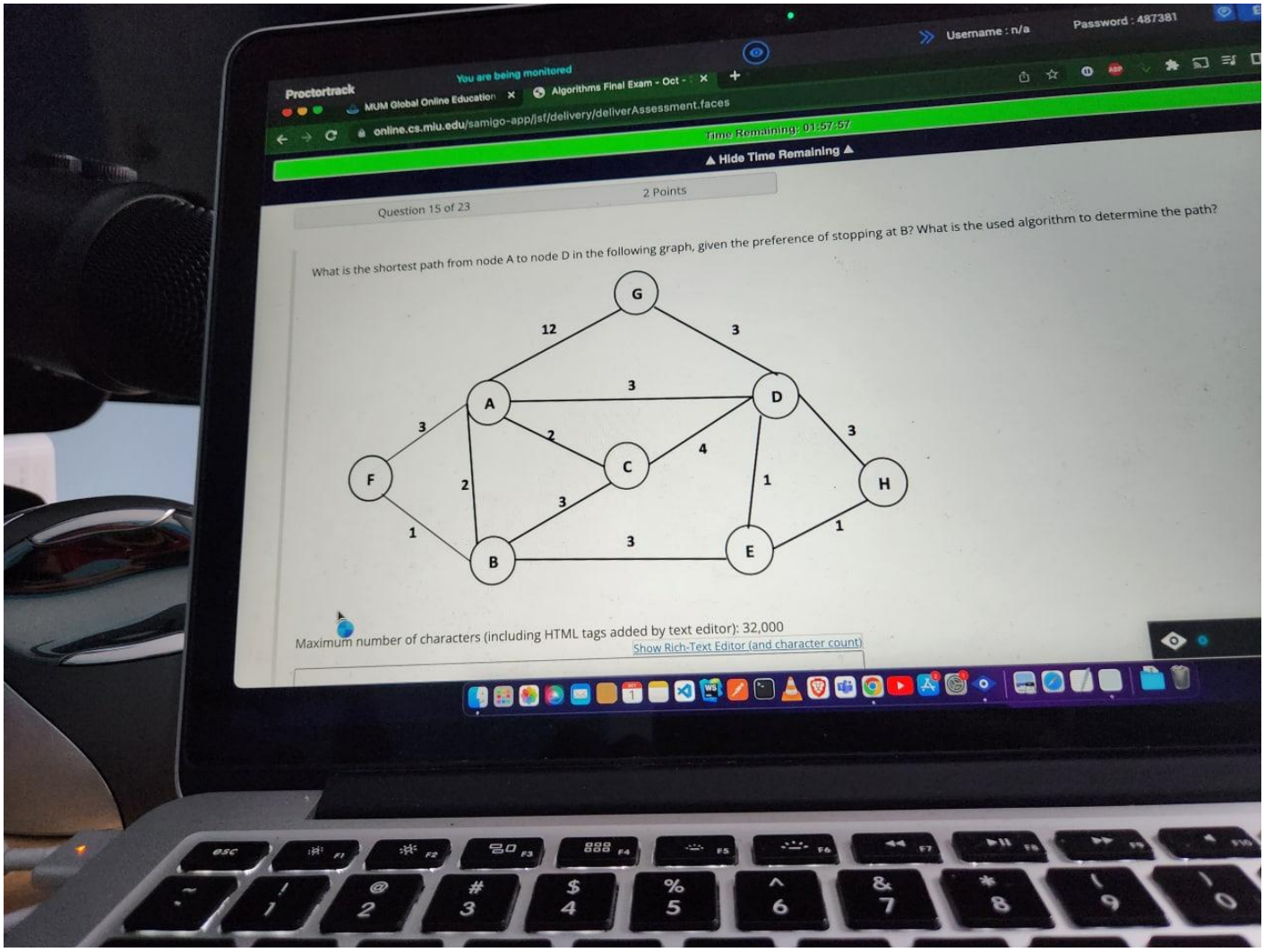


Question 14

Definition 1->A graph is a Hamiltonian graph if it contains a Hamiltonian cycle. In the mathematical field of graph theory, a Hamiltonian path (or traceable path) is a path in an undirected or directed graph that visits each vertex exactly once.

Definition-2->The graph will be known as a Hamiltonian graph if there is a closed walk in a connected graph, which passes each and every vertex of the graph exactly once except the root vertex or starting vertex. The Hamiltonian walk must not repeat any edge.

Answer. Yes this graph is a Hamiltonian graph and the possible cycle of this graph is as follows. 1-7-2-8-3-9-12-4-11-6-10-5-1



Question 15

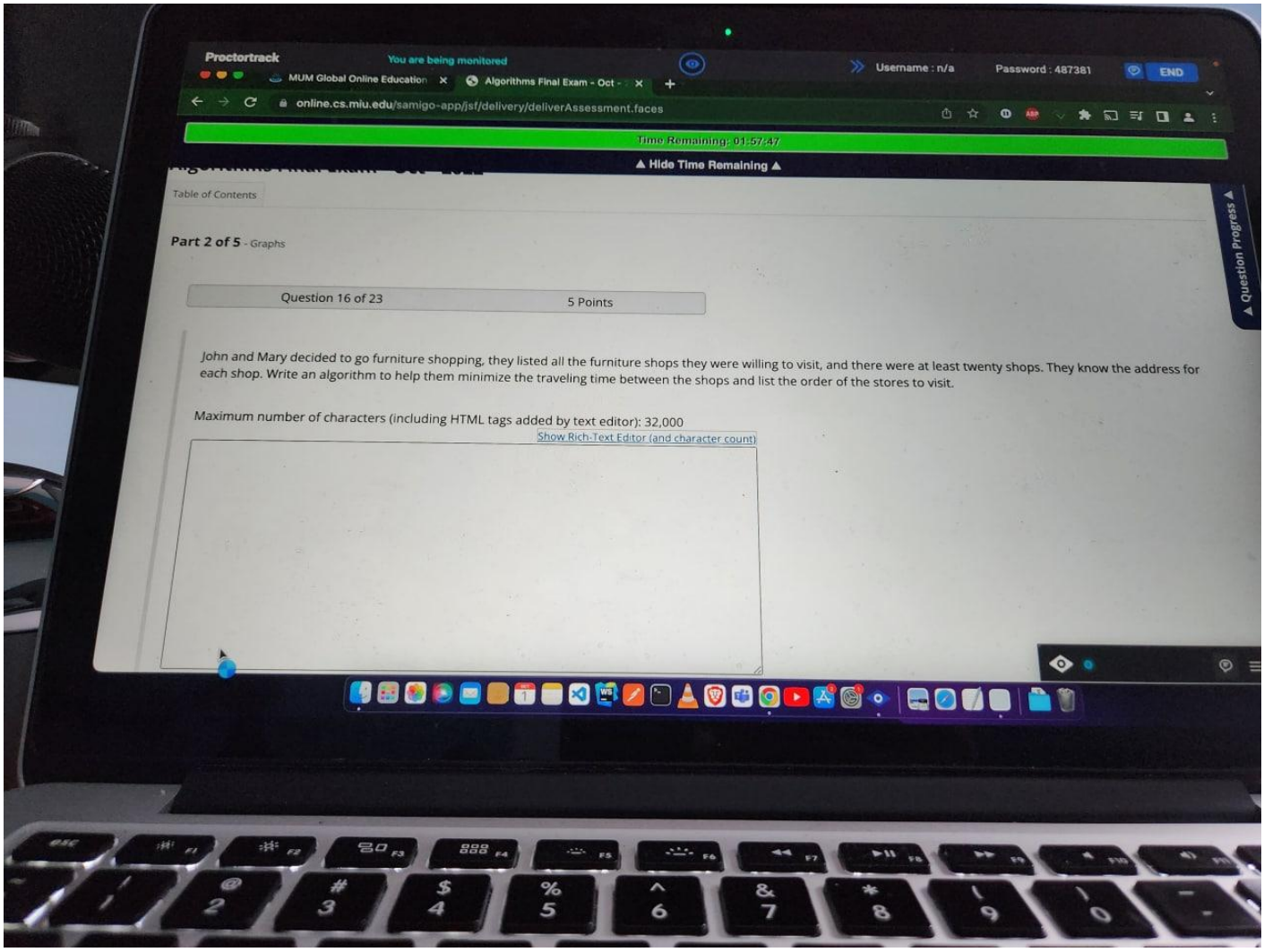
Answer

We use dijkstra shortest path algorithm for the weighted graph by finding the shortest from

A->B->E->D

A-B=2, B-E=3, E-D=1

2+3+1=6



Question 16
Answer

Algorithm: findMinTravelTime(shops)

Calculate distance between each shops
Visited -> create an array that will store visited nodes

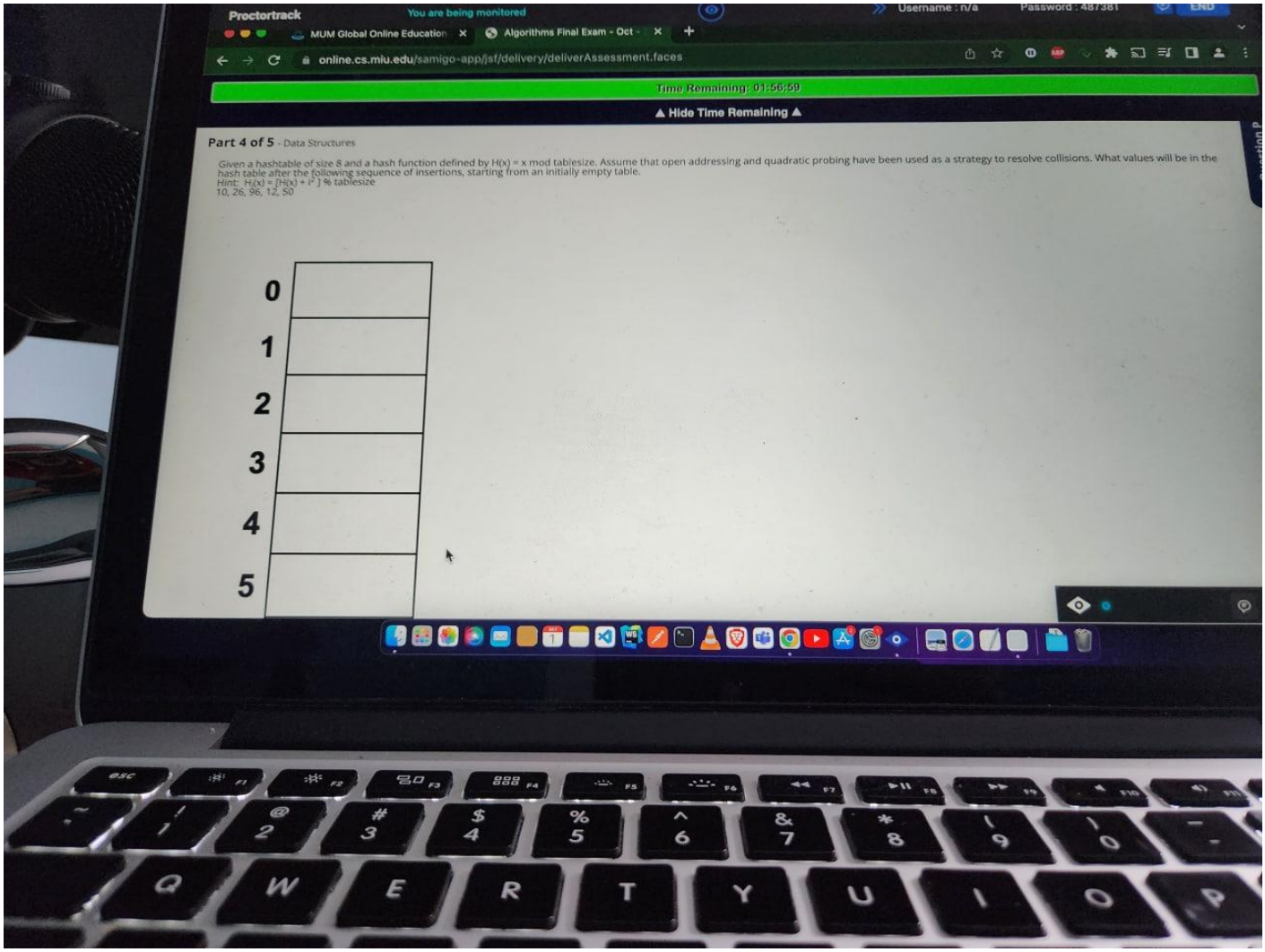
Choose minimum distance
Insert both shops (shop1, shop2) to Visted array

Shop3 = Select min distance connected to shop1 or shop2
Insert shop3 to visited array

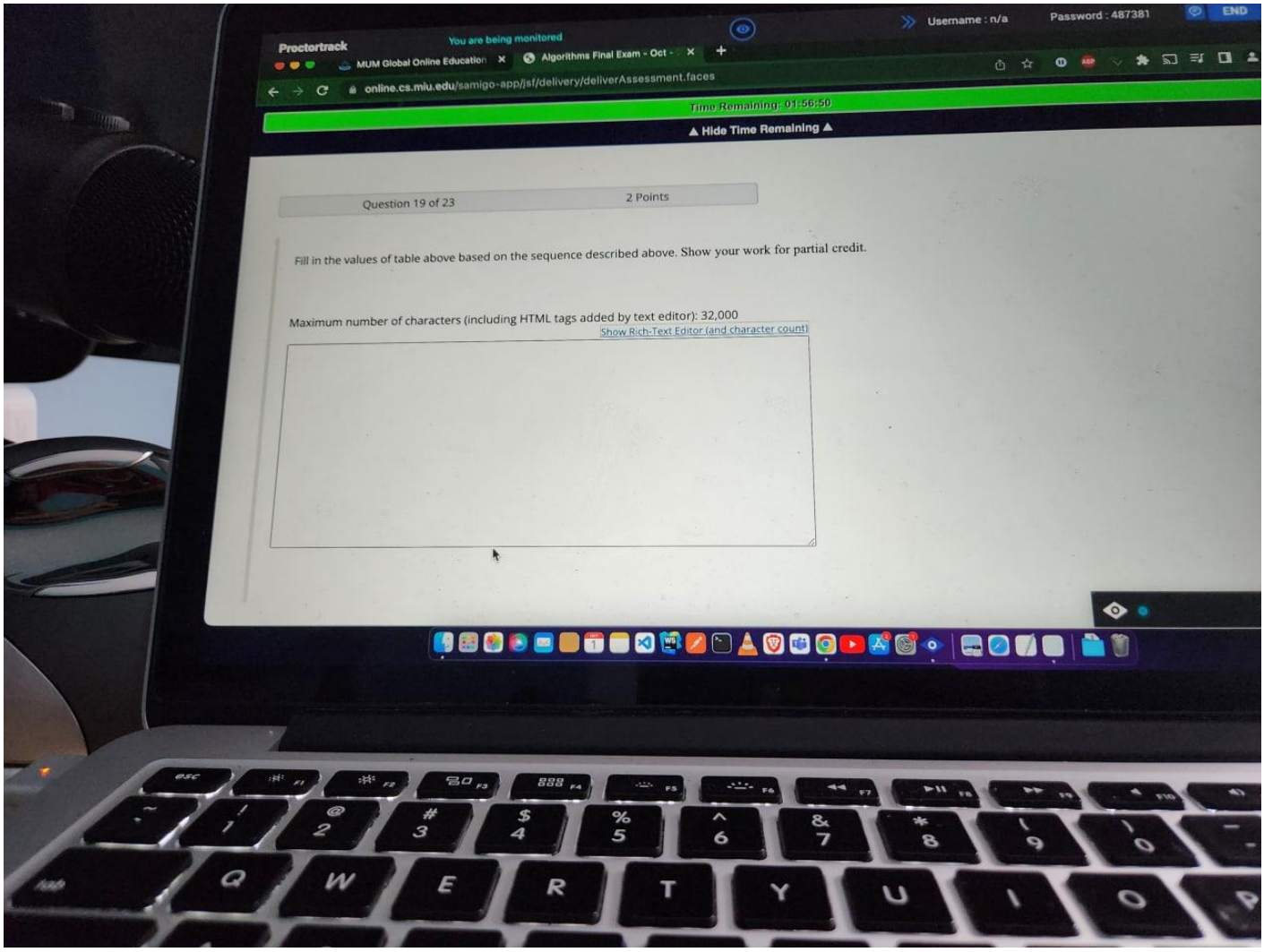
Let shopi = shop3
while there is unvested vertex existes do{
    Shopi = select min distance connected to shopi
    Shopi = move to another end of selected distance
}

Question 17
Answer
C
Rationale-when we insert the 87 to the red black tree, it will be inserted to the right of 87 as red, this time Red-red conflict will happen. When we check the sibling of the parent is black. Hence left rotation and recoloring will make it balanced. When we insert 88, still red-red conflict, but the sibling of the parent is red now. So recoloring is required. Then after we also recolored the grand parent. This time another red red conflict happens. The sibling is black, we need left rotation

Question 18
Answer:
The worst case running time complexity is O(n).
Example- Left and right skewed BST
One good solution to fix this problem would be to balance the tree as in AVL and Red black trees so that the running time becomes O(log n


Proctortrack
You are being monitored
Username : n/a
Password : 487381
MUM Global Online Education
Algorithms Final Exam - Oct -
online.cs.miu.edu/samigo-app/jsf/delivery/deliverAssessment.faces
Time Remaining: 01:56:59
▲ Hide Time Remaining ▲
Part 4 of 5 - Data Structures
Given a hashtable of size 8 and a hash function defined by H(x) = x mod tablesize. Assume that open addressing and quadratic probing have been used as a strategy to resolve collisions. What values will be in the hash table after the following sequence of insertions, starting from an initially empty table.
Hint: Hi(x) = [H(x) + i²] % tablesize
10, 26, 96, 12, 50
0
1
2
3
4
5

)



Question 19

Answer

Inputs: 10, 26, 96, 12, 50

0 → 96

1 →

2 → 10

3 → 26

4 → 12

5 →

6 → 50

7 →

For x = 10,

10 % 8 = 2

For x = 26,

26 % 8 = 2

Thus, we go to quadratic probing.

Index = (h(x) + i^2) % 8

i= 0, = (2 + 0) % 8 = 2 (not free)

i = 1, = (2 + 1) % 8 = 3 (free and hence we insert 26 at position 3)

For x = 96,

96 % 8 = 0

For x = 12,

12 % 8 = 4

For x = 50,

50 % 8 = 2

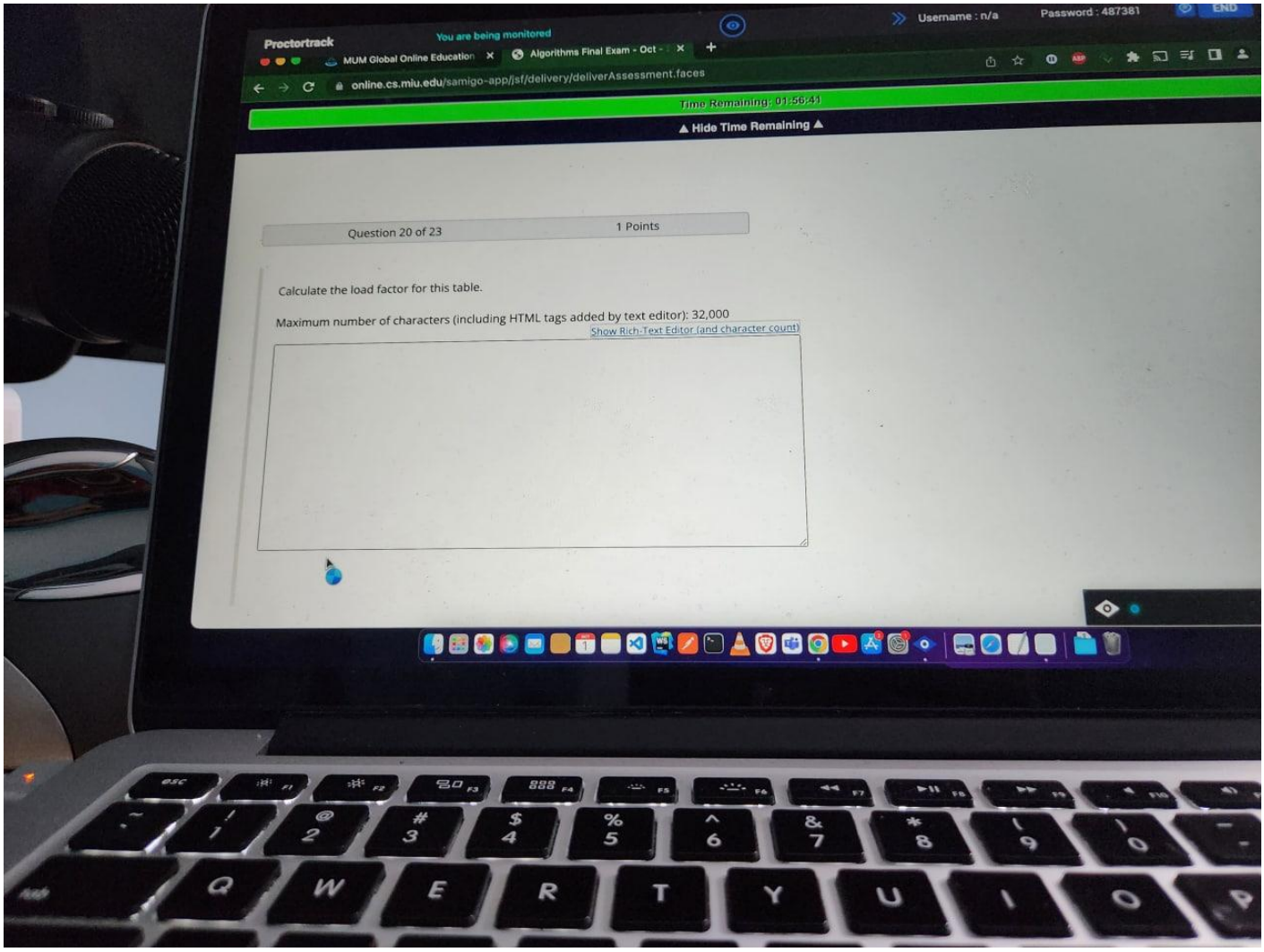
Thus, we go to quadratic probing.

Index = (h(x) + i^2) % 8

i= 0, = (2 + 0) % 8 = 2 (not free)

i = 1, = (2 + 1) % 8 = 3 (not free)

I = 2, = (2 + 4) % 8 = 6



Question 20 —- Corrected

Answer

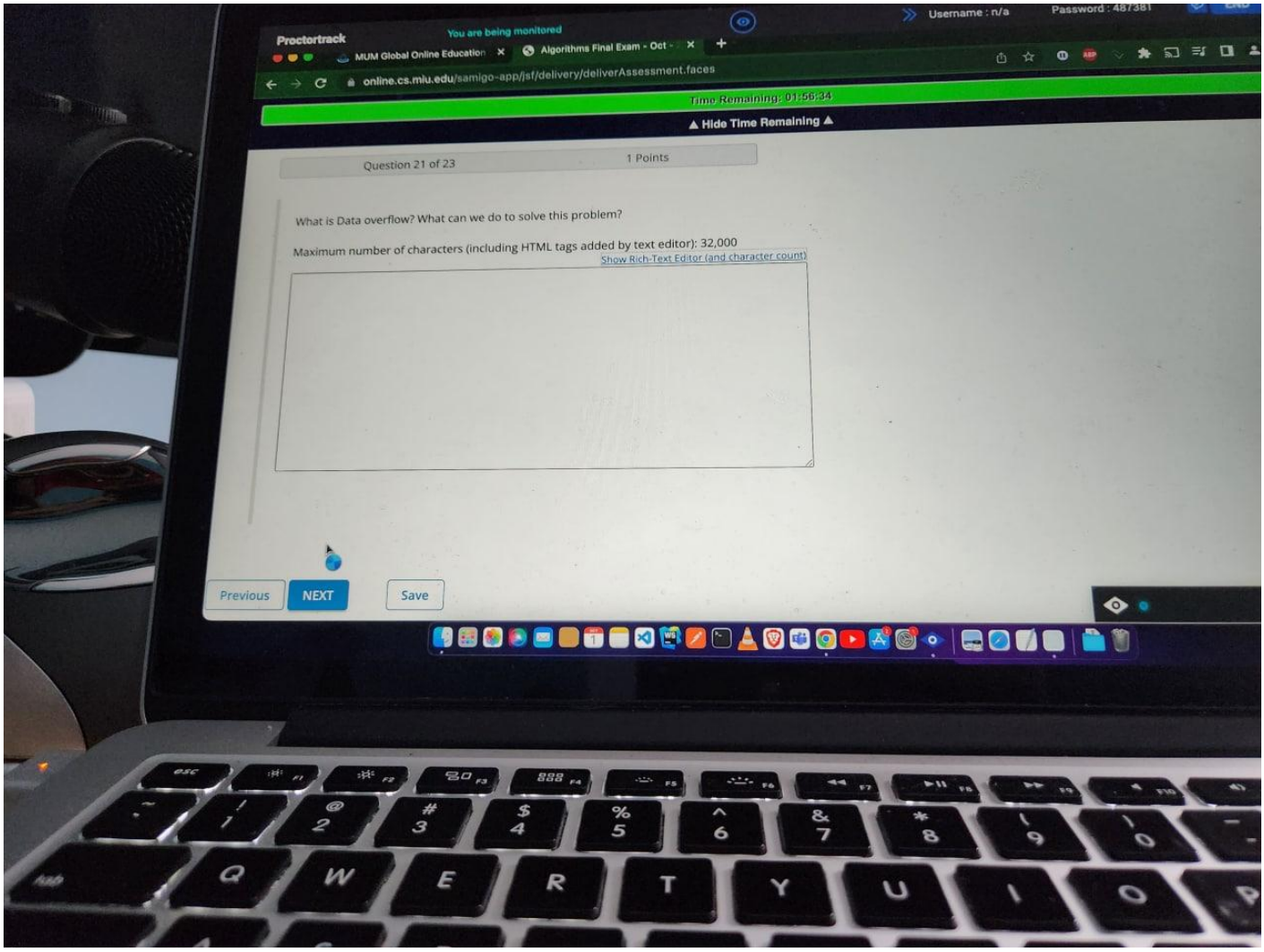
Load factor = number of elements / size

So

Number of elements =5

Size =8

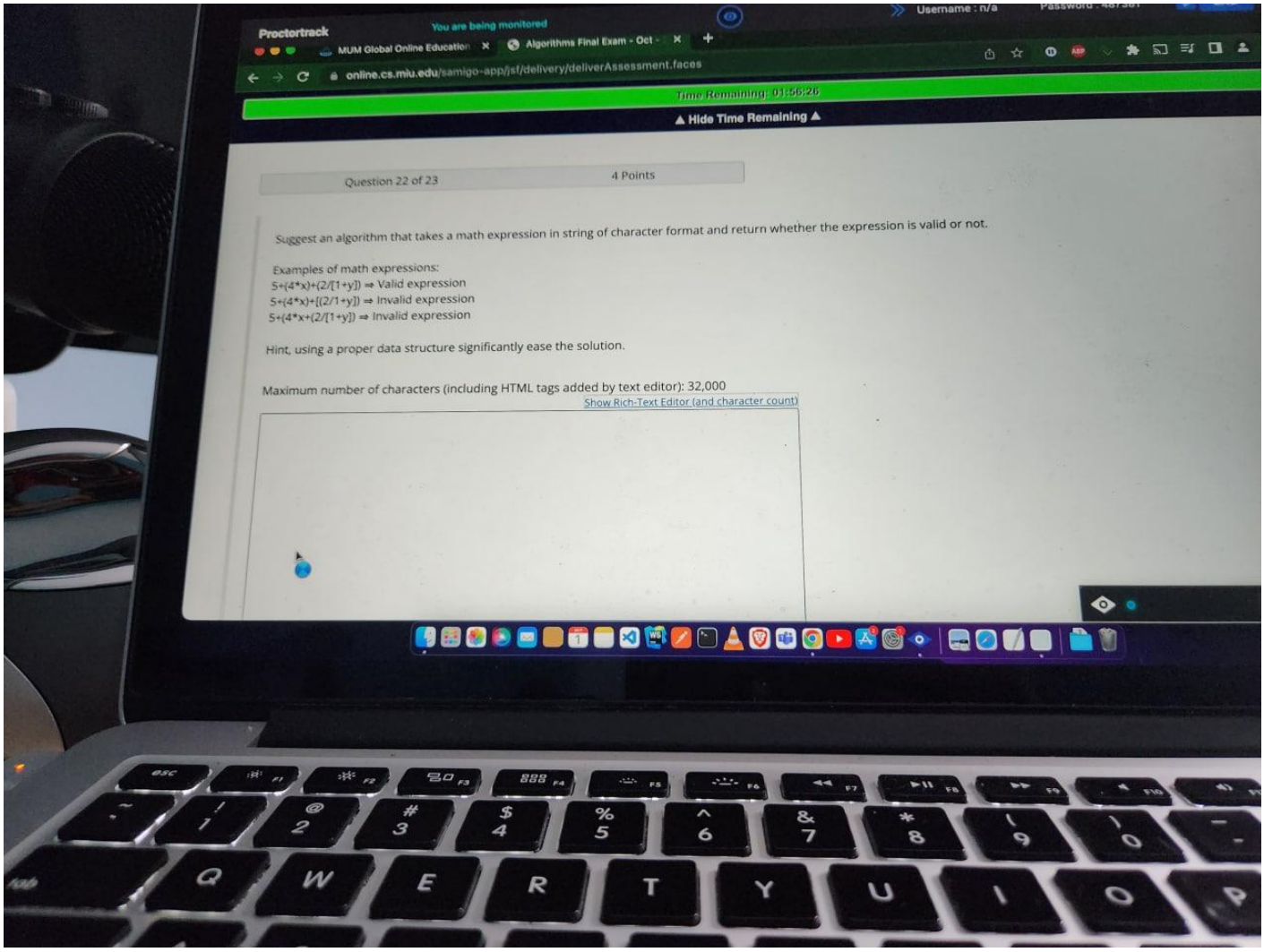
Load factor = 5 / 8 = 0.6



Question 21

Answer

Data overflow occurs when there is no empty slot is found in our bucket to hold data, it can be solved using rehashing by which we would create another larger size array and populate it by hashing the new array's keys again.



Question 22
Answer

We can use a Stack data structure along with a hashmap to solve this problem.

Algorithm checkValidString(String mathExpression)
Input: string of the mathematical expression
Output: true if valid else false

Create a HashMap<String, String> map = new Hashmap<>()
map.put("}", "{")

map.put(“]“, “[“)
map.put(“)”, “(“)

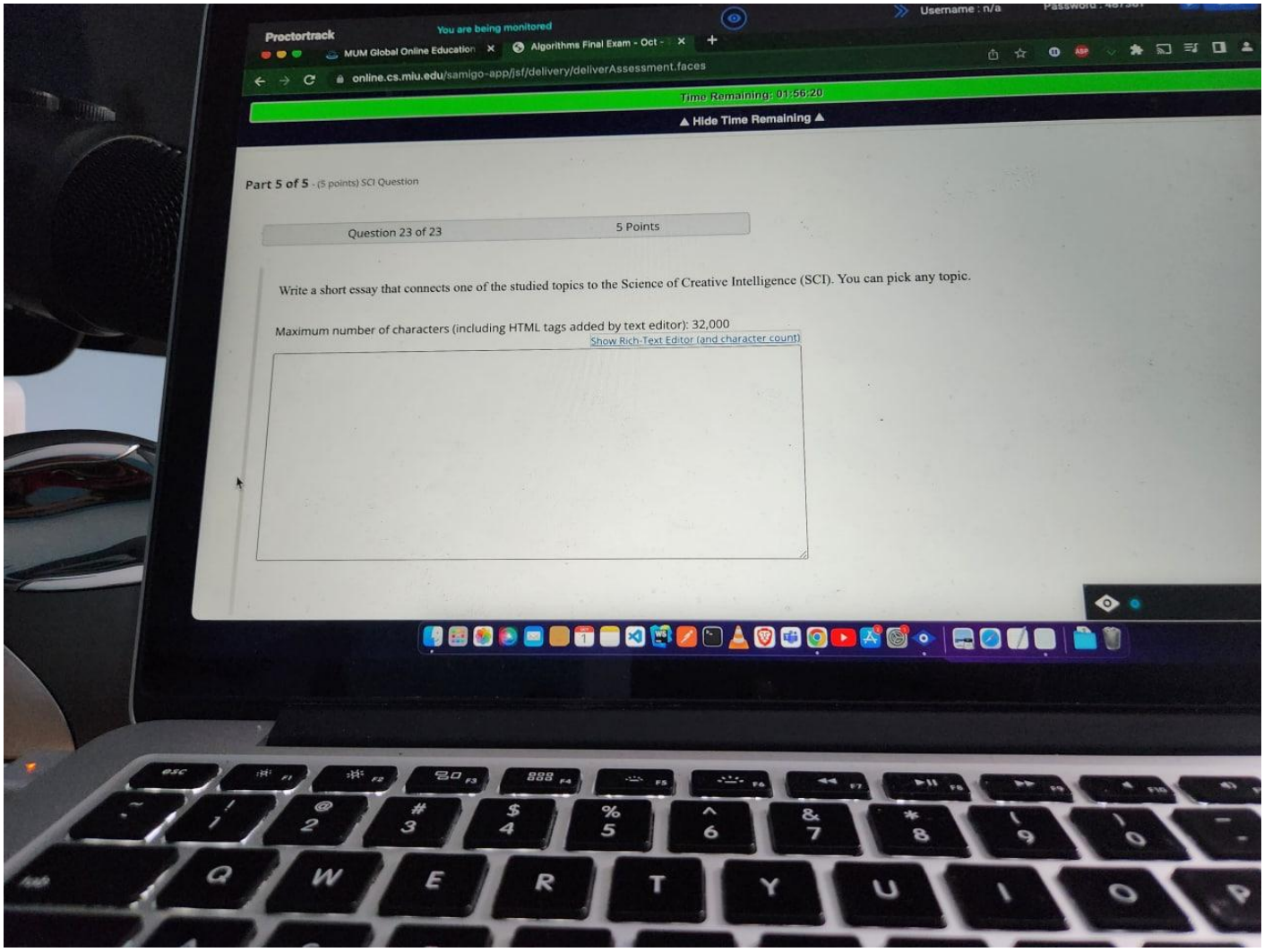
Create operatorStack and valueStack

while not end of String do
  Get next character string from the input string
if the character String is number or variable then
Push to valueStack
if character String is either “{“, “[“, or “(“ then
push it onto operatorStack.
if character String is “}”,”]” or “)” then {
if top of the operatorStack is not a “[“,”{“,”)” then{
pop the operator from the operatorStack.
pop the valueStack twice, getting two operands.
Apply the operator to the operands.
push the result onto the value stack.
        }
if map.get(character String) = value Pop from the operatorStack then
 Continue
  else
    return false;
}

if character String is an operator “*”,”+”, “-“or “/” then {
If operator stack is not empty and the operator in the top of the stack has
the same or greater precedence of current input operator then
Pop the operator from the operatorStack.
Pop the valueStack twice, getting two operands.
Apply the operator to the operands
Push the result onto the value stack.
Push current operator onto the operatorStack.

```
}
while operatorStack is not empty do
  Pop the operator from the operatorStack.
  Pop the valueStack twice, getting two operands.
  Apply the operator to the operands
  Push the result onto the value stack.

if operatorstack is empty and size(valueStack) = 1 then
  return true
else
  return false
```



Question 23

The hardest NP problems are NP-complete. These require the highest degree of creativity to solve. However, if a polynomial-time algorithm is found for any one of them, then all NP problems will automatically be solved in polynomial time. This phenomenon illustrates the fact that the field of pure consciousness, the source of creativity, is itself a field of infinite correlation – "an impulse anywhere is an impulse everywhere".