Madeleine Monfort
COP3530
12/1/15

MST

Testing:

Step 1) Taking in information
-linked list testing

Test	Output	Correct?
random inserting and deleting	printed correct order	Y

-testing edge

Test	Output	Correct?
check comparisons	if-statement produced correct result	Y
check printing	printed correctly	Y

-adjacency list testing (to represent the graph)

Test	Output	Correct?
make chain of edges	No errors	Y
insert edge into nothing	inserts edge	Y
print?	can print and prints out correctly	Y
insert multiple	inserts more edges	Y
erase one	gets rid of erased edge	Y
inserting something out of bounds	provides error message	Y
insert data from example	prints a reliable representation of graph	Y

Step 2) Prim's Algorithm

Task	how?	Complete?
add edges to heap	edges from each new "reached" node (found in the adjacency list)	Y
check if adding non-duplicate edges	if-statement, BEFORE? adding edge to heap	Y
hold values for reached edges	bool array (of nodes)	Y
hold value for weight of spanning tree	running sum int	Y

Task	how?	Complete?
output the spanning tree edges	each time edge identified, print it	Y
set bool of ifReached	each time edge identified, make new element have ifReached = true	Y

then ran the example and got correct result

Step 3) Kruskal’s Algorithm

Task	how?	Complete?
hold all edges	min heap	Y
make sure no duplicate edges	if nodeTo < nodeFrom, then it is a duplicate	Y
add edge	take least cost from heap	Y
preventing cycles	union find and adding a node each time an edge is “added”	Y
hold weights	running sum	Y
print edges	print as edge is “added”	Y

Testing…
-heap

Test	Output	Correct?
make an edge heap	no error	Y
insert edges into heap	inserts without error	Y
print edges	prints in correct format and order of least to greatest weight	Y
top	returns smallest weighted edge	Y

-Union Find

Test	Output	Correct?
union ints	NA	Y
check “connected”	outputs correct bool	Y

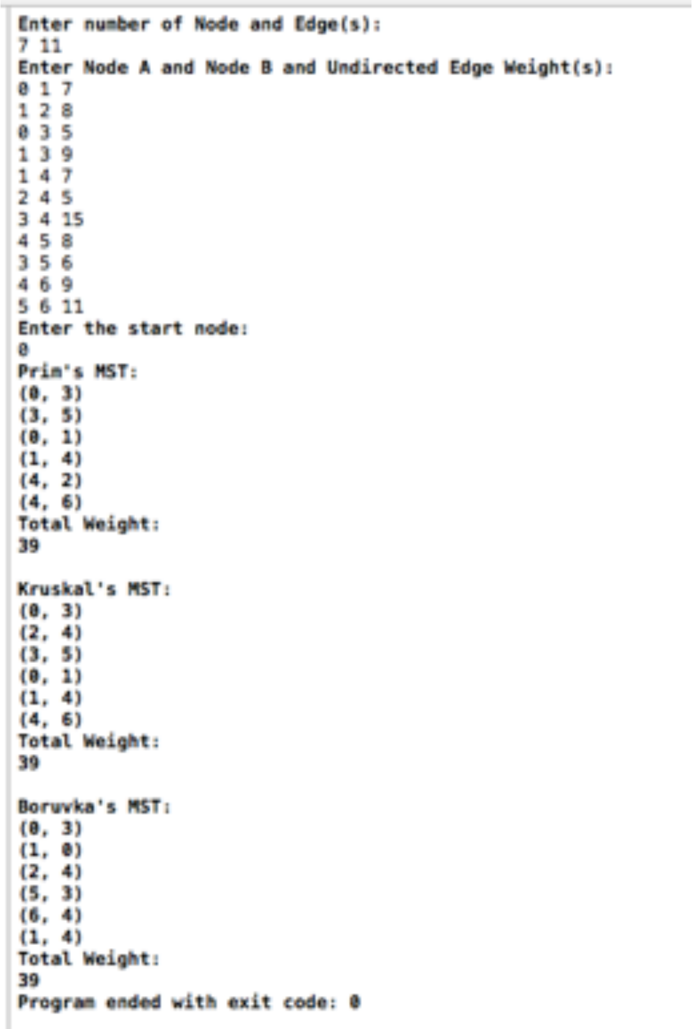
ran algorithm on example and got correct output

Step 4) Sollin’s Algorithm

Task	**how?**	**Complete?**
hold components	array of nodes, the value in the array is the “head” of the component the index is in	Y
vertex in component	for loop for components, while the value is the same	Y
find smallest edge	heap for each component	Y
check that is not in component	use array of components	Y
add edges each time it is necessary	chain of edges to hold final edges	Y
make while loop end	end when an int arrives at n-1 (increment the int every time edge is added)	Y
print edges	print at end from edge chain	Y
total weight	running sum	Y

To TEST: ran algorithm on example and got correct output

TOTAL TEST:
tested on the first graph given. Got correct results: *Note: only integers can be used for Nodes



```
Enter number of Node and Edge(s):
7 11
Enter Node A and Node B and Undirected Edge Weight(s):
0 1 7
1 2 8
0 3 5
1 3 9
1 4 7
2 4 5
3 4 15
4 5 8
3 5 6
4 6 9
5 6 11
Enter the start node:
0
Prim's MST:
(0, 3)
(3, 5)
(0, 1)
(1, 4)
(4, 2)
(4, 6)
Total Weight:
39

Kruskal's MST:
(0, 3)
(2, 4)
(3, 5)
(0, 1)
(1, 4)
(4, 6)
Total Weight:
39

Boruvka's MST:
(0, 3)
(1, 0)
(2, 4)
(5, 3)
(6, 4)
(1, 4)
Total Weight:
39
Program ended with exit code: 0
```