# Stations

## Searching

Initialize a node pointer, current = head.

Do following while current is not NULL

 If the current value (i.e., current->key) is equal to the key being searched return true.

Otherwise, move to the next node (current = current->next).

If the key is not found, return false

BEGIN

Declare search

Declare found=false

Declare Station\*node

\*node=head

While(node!=null)

If(search==node->station\_name)

Found=true

Display node-> station\_ID

Display node-> station\_name

Display node->

Display node->

Display node->

Display node->

ENDIF

Node=node->next

ENDWHILE

END

# Transactions

## Searching

Step 1 : Initialize, start\_node (head of list) and last\_node (will have last value) , mid\_node (middle node of the structure).

Step 2 : Compare mid\_node to element

   Step 2.1 : if mid\_node = element, return value “found”.

   Step 2.2 : if mid\_node > element, call binary search on lower\_Half.

   Step 2.3 : if mid\_node < element, call binary search on upper\_Half.

Step 3 : if entire list is traversed, return “Not found”.

BEGIN

Declare search\_value

Display “ Enter search\_value”

Read search

Initialize start\_node

Initialize last\_node

DOWHILE last\_node == NULL and last\_node != start\_node

Declare Transaction\*node

\*node=head

Nod\*mid = middle(start\_node, last\_node)

If mid==NULL THEN

Return mid

Elseif mid->node== search\_value THEN

Display node-> Transaction\_ID

Display node-> Ticket\_ID

Display node-> Source\_station

Display node-> Natargert\_station

Display node-> total\_amount

Display node-> Date\_Time

Display node-> Departure\_time

Display node-> Customer\_ID

Display node-> Customer\_name

Display node-> Identity Card

Display node-> Passport

Elseif mid->node < search\_valeue THEN

start\_node = mid -> next;

elseif

last\_node = mid;

else

Display “Not found”

ENDIF

ENDWHILE

END

## Sorting

Based on https://www.geeksforgeeks.org/merge-sort-for-linked-list/

FUNCTION sort\_transactions(transactions)

BEGIN

DECLARE head = transactions

IF head = NULL OR head.next = NULL THEN

END FUNCTION

ENDIF

DECLARE mid = head

DECLARE end = head.next

DOWHILE end != NULL

end = end.next

IF end != NULL THEN

mid = mid.next

end = end.next

ENDIF

ENDDO

mid.next = NULL

sort\_transactions(head)

sort\_transactions(mid)

transactions = merge(head, mid)

END

FUNCTION merge(a, b)

BEGIN

DECLARE result = NULL

IF a = NULL THEN

RETURN b

ELSE

IF b = NULL THEN

RETURN a

ENDIF

ENDIF

IF a.date <= b.date THEN

result = a

result.next = merge(a.next, b)

ELSE

result = b

result.next = merge(a, b.next)

ENDIF

RETURN result

END