# Stations

## Searching

|  |
| --- |
| BEGIN​  Declare inputName​  Display “Enter station name”​  Read inputName​  Declare found=false​  Declare Station\*node​  \*node=head​  While(node!=null)​        If(node->station\_name ==inputName)​                Found=true​                Display Station\*node​  Else ​          Display “ station not found”​  Node=node->next ​      ENDIF​  ENDWHILE​  END​  ​ |

# Transactions

## Searching

|  |
| --- |
| FUNCTIONsearch\_passenger(arr, length, passenger)​  BEGIN​  IF length = 0 THEN​      RETURN NULL​  ENDIF​  DECLARE mid = length/2​  IF passenger =arr[mid].name THEN​      RETURN arr[mid]​  ENDIF​  IF passenger <arr[mid].name THEN​      RETURNsearch\_passenger(arr, mid, passenger)​  ENDIF​  RETURNsearch\_passenger(arr[mid], mid, passenger)​  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