**Functions**

This document contains all the functions used to build the London Tube Navigation Expert System. The Table below is linked to specific sections in the document for easy navigation.

|  |  |
| --- | --- |
| **Function Name** | **Description** |
| [find-index](#_Function_–_find-index) | Calculates the position of a station in the before and after list of another station |
| [bsplice](#_Function_–_bsplice) | Calculates the subset of stations from before station list that need to be travelled to get to destination |
| [asplice](#_Function_–_asplice) | Calculates the subset of stations from the after station list that need to be travelled to get to destination |
| [calc\_cost](#_Function_–_calc_cost) | Calculates the cost of travel from one station to another in route based on zone positioning |

## Function – find-index

;;-----------------------------------------------------

;; Function: find-index

;; Description: This function is designed to find the index of a specified element within a multislot.

;; Parameters:

;; - ?element: The element to search for within the multislot.

;; - ?multislot: The multislot in which the search for the element is conducted.

;; Returns:

;; - The index of the first occurrence of the specified element in the multislot. Returns 0 if the element is not found.

;;-----------------------------------------------------

(deffunction find-index (?element ?multislot)

(bind ?l (length$ ?multislot)) ;; Get the length of the multislot

(bind ?index 1) ;; Initialize the index to 1

(loop-for-count ?l do ;; Loop over the elements of the multislot

(if (eq ?element (nth$ ?index ?multislot))

;; Check if the current element is equal to the specified element

then

(return ?index) ;; Return the index if the element is found

)

(bind ?index (+ ?index 1)) ;; Increment the index for the next iteration

)

(return 0) ;; Return 0 if the element is not found in the multislot

)

## Function – bsplice

;;-----------------------------------------------------

;; Function: bsplice

;; Description: Calculates the subset of stations from the station list before the specified index.

;; Parameters:

;; - ?b1: The original station list.

;; - ?index: The index specifying the subset of stations to be included (inclusive).

;; Returns:

;; - The subset of stations from the original list up to the specified index.

;;-----------------------------------------------------

(deffunction bsplice(?b1 ?index)

(bind ?l (length$ ?b1)) ;; Get the length of the original station list

(bind ?n 1) ;; Initialize a counter for looping through the station list

(loop-for-count ?l do ;; Loop over the elements of the station list

(if (>= ?n ?index)

;; Check if the current index is greater than or equal to the specified index

then

(if (= ?n ?index) then (bind $?lst (nth$ ?n ?b1))

;; If the current index is equal to the specified index, bind the current station to $?lst

else (bind ?lst (nth$ ?n ?b1) ?lst)

;; If the current index is greater than the specified index, append the current station to ?lst

)

)

(bind ?n (+ ?n 1)) ;; Increment the counter for the next iteration

)

(return ?last) ;; Return the subset of stations up to the specified index

)

## Function – asplice

;;-----------------------------------------------------

;; Function: asplice

;; Description: Calculates the subset of stations from the station list after the specified index.

;; Parameters:

;; - ?a1: The original station list.

;; - ?index: The index specifying the subset of stations to be included (inclusive).

;; Returns:

;; - The subset of stations from the original list starting from the specified index.

;;-----------------------------------------------------

(deffunction asplice(?a1 ?index)

(bind ?l (length$ ?a1)) ;; Get the length of the original station list

(bind ?n 1) ;; Initialize a counter for looping through the station list

(loop-for-count ?l do ;; Loop over the elements of the station list

(if (<= ?n ?index)

;; Check if the current index is less than or equal to the specified index

then

(if (= ?n 1) then (bind $?lst (nth$ ?n ?a1))

;; If the current index is 1, bind the current station to $?lst

else (bind ?lst ?lst (nth$ ?n ?a1))

;; If the current index is greater than 1, append the current station to ?lst

)

)

(bind ?n (+ ?n 1)) ;; Increment the counter for the next iteration

)

(return ?lst) ;; Return the subset of stations starting from the specified index

)

## Function – calc\_cost

;;-----------------------------------------------------

;; Function: calc\_cost

;; Description: Calculates the cost of travel from one station to another based on zone positioning.

;; Parameters:

;; - ?st1: The name of the starting station.

;; - ?st2: The name of the destination station.

;; Returns:

;; - The cost of travel between the two stations based on their zone positioning.

;;-----------------------------------------------------

(deffunction calc\_cost(?st1 ?st2)

(bind $?Station1 (find-all-facts ((?s1 Station)) (eq ?s1:name ?st1)))

;; Find all facts for the starting station

(bind $?Station2 (find-all-facts ((?s2 Station)) (eq ?s2:name ?st2)))

;; Find all facts for the destination station

(bind $?zone1 (fact-slot-value ?st1 zone)) ;; Get the zone value for the starting station

(bind $?zone2 (fact-slot-value ?st2 zone)) ;; Get the zone value for the destination station

(if (eq ?zone1 ?zone2) then

;; Check if the starting and destination stations are in the same zone

(if (eq (length$ ?zone1) 1) then ;; Check if the zone has a single value

(if (eq (nth$ 1 ?zone1) 1) then (return 2.4) ) ;; If the zone value is 1, return a cost of 2.4

(if (eq (nth$ 1 ?zone1) 2) then (return 2.9) ) ;; If the zone value is 2, return a cost of 2.9

else (return 2.4) ;; If the zone has multiple values, return a default cost of 2.4

)

else ;; If the starting and destination stations are in different zones

(if (eq (length$ ?zone1) 1) then ;; Check if the starting station has a single zone value

(return 2.9) ;; If it does, return a cost of 2.9

else (return 2.4) ;; If it has multiple zone values, return a default cost of 2.4

)

)