Traffic Signal System

Function Description

1. **General Functions**
   1. **Set ID**
      * **Syntax:**

**char \*set\_id(char \*parent\_id,char \*type\_id);**

* + - **Description:**
      * The function sets the id for an object.
      * It would take the parent’s ID as a parameter.(Pass “”{BLANK} as a parameter) and the type of the object (as a string).
      * *NOTE:* Refer to the Documentation Procedures
    - **Parameter : char \*parent\_id**(string), and type (string)
    - **Return Type : char \***(string)
  1. **Get ID Number :**
     + **Syntax :**

**int get\_id\_no( char \* id);**

1. **Traffic Signal Light (TSL)**
   1. **Create Traffic Signal Light** 
      * **Syntax:**

**struct traffic\_light \*create\_tsl(void);**

* + - **Description:**
      * The function would create a TSL by dynamically allocating the memory.
      * It would *return* a pointer to the created TSL.
    - **Return Type:** traffic\_light \* (pointer to traffic\_light)
  1. **Create n Traffic Signal Lights** 
     + **Syntax:**

**struct traffic\_light \*\*create\_n\_tsl(int n);**

* + - **Description:**
      * The function would create *n* TSL by dynamically allocating the memory.
      * It would *return* a double pointer to the created TSLs.
    - **Return Type:** traffic\_light \*\* (double pointer to traffic\_light)
  1. **Set TSL Mode [ set\_tsl\_mode() ]**
     + **Syntax:**

**int set\_tsl\_mode(struct traffic\_light \*tsl, int mode);**

* + - **Description:**
      * This function would contain multiple cases of all possible combinations of the operation of a single traffic signal light at a junction.
      * It would take ***mode*** as a parameter. Each *mode* would correspond to a particular setting of the traffic signal light for a particular lane incoming at a traffic signal junction.
      * It would return the SUCCESS(1) or FAILURE(0) to set mode.
    - **Parameters :**

|  |  |  |
| --- | --- | --- |
| **Name** | **Data type** | **Description** |
| mode | int | Each *mode* would correspond to a particular setting of the traffic signal light for a particular lane incoming at a traffic signal junction. |

* + - **Return Type**: boolean(int 1/0)

1. **Lane** 
   1. **Create Lane [ create\_lane()]**
      * **Syntax:**

**struct lane \*create\_lane(int front, int rear, );**

* + - **Description:**
      * The function would create a *lane* by dynamically allocating the memory.
      * It would *return* a pointer to the created lane.
    - **Return Type: struct** lane \* (pointer to lane)
  1. **Create n Lanes [ create\_n\_lane()]**
     + **Syntax:**

**struct lane \*\*create\_lane( int n );**

* + - **Description:**
      * The function would create n *lanes* by dynamically allocating the memory.
      * It would *return* a double pointer to the created lanes.
    - **Return Type:** struct lane \*\* (double pointer to lane)
  1. **Create Queue[ create\_queue()]**
     + **Syntax:**

**vehicle\_t \*\* create\_queue( int capacity );**

* + - **Description:**
      * The function would create a queue
      * It would *return* a double pointer to the created lanes.
    - **Return Type:** struct lane \*\* (double pointer to lane)
  1. **Add Vehicle**
     + **Syntax:**

**int queue\_add\_vehicle( lane\_t \*l,vehicle\_t \*v );**

* + - **Description:**

**Street**

* 1. **Create Street [ create\_street()]**
     + **Syntax:**

**struct street \*create\_street(char \*name, char \*type, int max\_vehicles, int curr\_vehicles, int timer );// add timer from stimulation function**

* + - **Description:**
      * The function would create a *street* by dynamically allocating the memory.
      * Each street will also create the lanes leading from one
      * It would *return* a pointer to the created *street*.
    - **Return Type:** struct street \* (pointer to street)

1. **Map Extension Node**
   1. **Create Map Extension [ create\_map\_extn()]**
      * **Syntax:**

**struct map\_extn \*create\_map\_extn(struct traffic\_signal\_jn \*tsj );**

* + - **Description:**
      * The function would create a *map extension node* by dynamically allocating the memory.
      * The parameter *tsj* will set the adjacent TSJ node to the map extension node.
      * It would *return* a pointer to the created *map extension node*.
    - **Return Type:** struct street \* (pointer to street)

1. **Traffic Signal Junction (TSJ)**
   1. **Create TSJ**
      * **Syntax:**

**struct traffic\_signal\_jn \*create\_tsj(void);**

* + - **Description:**
      * The function would create a TSL by dynamically allocating the memory.
      * It would *return* a pointer to the created TSL.
    - **Return Type:** traffic\_signal\_jn \*(pointer to traffic\_signal\_jn)
  1. **Pop -up on crossing red light // part of stimulator**
  2. **Analyse Traffic Signal Junction [analyse\_jn()]**
     + **Syntax:**

**int analyse\_jn (struct traffic\_signal\_jn \*tsj);**

* + - **Description:**

This function would analyze the traffic level on each lane connected to the traffic signal junction (TSJ) and accordingly set the TSL mode for teach TSL corresponding to each lane.

* + - * Each type of junction would have different meanings for different modes. In this project, we are considering three types of junctions, i.e. 3, 4 or 5 lanes connected at a junction. Thus different outcome would have to be constructed for each type of junction.
    - **Return Type**: boolean(int 1/0)
    - **Requirements:**
      * List out all practical situations of the traffic signal that are possible for each of the 3,4 or 5 lanes connected to a traffic signal junction along with the mode for each of the TSL’s.
  1. **Creating Mode for traffic junction depending on the type of the street.**