Car Parking System in India  
(Algorithm)

1. First we declared the required header files.
2. Then, we declared a structure named ‘Parking’ that holds the value - that is the car number and the left and right pointers of the Binary Search Tree(BST).
3. The objects of the structures are –

* **Root**, that will help in creating the BST.
* **Temp**, that will help in increasing the BST Elements.
* **T1** help us in finding the element with 0 or 1 child for deletion and **T2**, the greatest and least elements in the BST and 2 child elements for deletion.
* **Left and right pointers** that are mentioned in the parking structure helps in choosing the left and the right path.

1. We declare the required prototypes for the program.
2. In the main function, we ask the user for the password, if it is correct, then we go for the menu driven program, and if it is wrong then we stop the program.
3. Then we start making the parking lot using the BST.
4. First we go for insertion. The insertion of cars involve three functions –

* **insert()** – To insert a node into a tree
* **create()** – To create a node
* **search(struct Parking\*t**) – To search for another node position if the root node is present.

1. Next, we go for removal of cars. This involves five functions:

* **delete()** - To delete a node from a tree.
* **search1(struct Parking\*t, int data)** – To search for the position of the desired root to be deleted.
* **delete1(struct Parking\*t)** – To delete the node, it can be of zero, one or two children.
* **smallest(struct Parking\*t)** – To find the smallest node on the left.
* **largest(struct Parking\*t)** – To find the largest node on the right.

1. Next, we go for traversal of the parking lot in ‘inorder’ manner, using the function **display(struct Parking\*t)** (NOTE: We are using inorder, because it is the most convenient way to print the elements).
2. Now, we will check how many cars are there in the parking lot. For this, we are using a counter and, according to the counter number, at a particular instance of time, we are getting the number of cars in the parking lot.
3. Now, we consider that the operations of the BST are over, and hence we move to Queue Operations.
4. Now, the parking lot is occupied, and this is considered to be the size of parking lot. The bulking of the cars outside the parking lot, is considered, that is the operation of queue is implemented now.
5. First, we enter the number of cars, which have come at a particular point of time, in the queue.

* **enq(char data[100])** – This helps in entering the car numbers in the queue, and this in return would be put in the BST, into the positions where spaces will be encountered.

1. In order to check the car which has come first into the queue, we call the following function

* **frontelement()** – This gives us the car at the first position.

1. We get to know the number of cars present in the queue by the help of the function **queuesize().**
2. Now, if we want to go back to the BST, press the appropriate option, and press enter.
3. Now, back in BST, after all the operations in the Queue and the BST are done satisfactorily, we exit from the program printing – **“Thank you for visiting!”** and exit.

Thank you!