Programming Assignment 4
The Jurassic Park Problem
CO21BTECH11004

In output file time is printed in microseconds which is the difference between time measured at that instant and start time.

Global Variables: -

- int/long
  - P: - number of passengers
  - C: - number of cars
  - k: - number of ride request by each passenger
  - passengerInMuseum: - have value 0 if all threads exit else 1
  - *car_track: - array that tracks which car rides which passenger, has passenger_id at the ith index (ith car) else 0.
  - *passengerTime: - to store time taken by passenger to complete tour
  - *carTime: - to store total ride time of a car on a tour.
- File *outFile: - global pointer to output array
- sem_t
  - *passenger_sem: - to wait the passenger thread until the car finishes the ride.
  - binSem: - for atomic updation of car_track array.
  - carAvail: - to ensure passenger request for car is only accepted if car is available

Function declared: -

- passenger(long passenger_id)
  - Print entry time in output file
  - Sleep passenger thread for random time (wander time).
  - loop is initialized for k times
    - Print request time of passenger in output file
    - Check if car is available (sem: - carAvail))
    - If available, then update car track array
    - Prints which car accepts request in output file
    - Prints passenger’s ride start time in output file

        - Wait for car to ride passenger (sem: - passenger_sem)
        - Prints passenger’s ride finish time
        - Passenger sleeps for time between two successive rides.
    - Update passengerTime array with passenger tour time in museum.
    - Prints exit time in the output file

- car(long thread_id): -
  while loop so threads runs until all passengers exits
    - If no passengerInMuseum breaks the while loop.
    - Waits for passenger (car_track[car_id-1]==0)
    - If passenger have updated car_track, print car is riding the passenger in output file
    - Sleep for random time (ride time).
    - Print car has finished riding the passenger
    - Update car ride time in carTime array.
    - signal passenger thread, that ride is done.
    - sleep for some time. (time between 2 successive request accepted by car)
    - signal carAvail.

main(): -
- Open the input file and take input.
- Initialize and dynamically allocate memory to the respective global variable.
- Create a passengerId and carId array.
- Create P passenger threads, C car threads.
- Join P passenger threads, set passengerInMuseum to 0, join car threads.
- Calculates the average time for passenger and total ride time of car for the tour.
- Destroy sem and free memory.

Design: -
- To ensure no more C accepts of passengers are accepted by car at a time, semaphore carAvail is used.
    - Sem_t carAvail is initialized to C.

    - sem_wait(&carAvail), is called by a passenger after making a request, it ensures only if any car is available then its request is accepted.
    - After the car finishes the passenger ride it signals carAvail semaphore.
- To check which car is free, passenger iterates through the carTrack array, to ensure only one passenger thread at a time iterates through it, binSem semaphore is used.
    - Sem_t binSem is initialized to 1.
    - If a passenger wants to access the carTrack array it calls sem_wait(&binSem) and after finding the car updates the array and signals binSem semaphore.
- To wait the passenger thread till the car is riding the passenger, a semaphore array passenger_sem is used.
    - Sem_t *passenger_sem is dynamically allocated and each semaphore is initialized to 0.
    - After updating the carTrack array, the car starts to ride the passenger, so in the passenger thread function, sem_wait(&passenger_sem[passenger_id-1]), so only that particular passenger thread waits.
    - After the car has finished riding the passenger it signals the passenger_sem semaphore, and it get the passenger_id by carTrack array.

Implementation of Semaphore: -
Library use: - semaphore.h

Semaphore declaration:-

```
// define semaphore
sem_t *passenger_sem;
sem_t binSem,carAvail;
```

Semaphore initialization: -

```
// initialize binSem and carAvail
   sem_init(&binSem,0,1);
   sem_init(&carAvail,0,C);
```

```
// dynamically allocate memory for counting semaphores
   passenger_sem = (sem_t *)malloc(P*sizeof(sem_t));

   // initialize counting semaphore
   for (i = 0; i < P; i++) sem_init(&passenger_sem[i],0,0);
```

Semaphore wait: -

```
sem_wait(&carAvail);
```

Semaphore signal: -

```
      // signal carAvail
       sem_post(&carAvail);
```

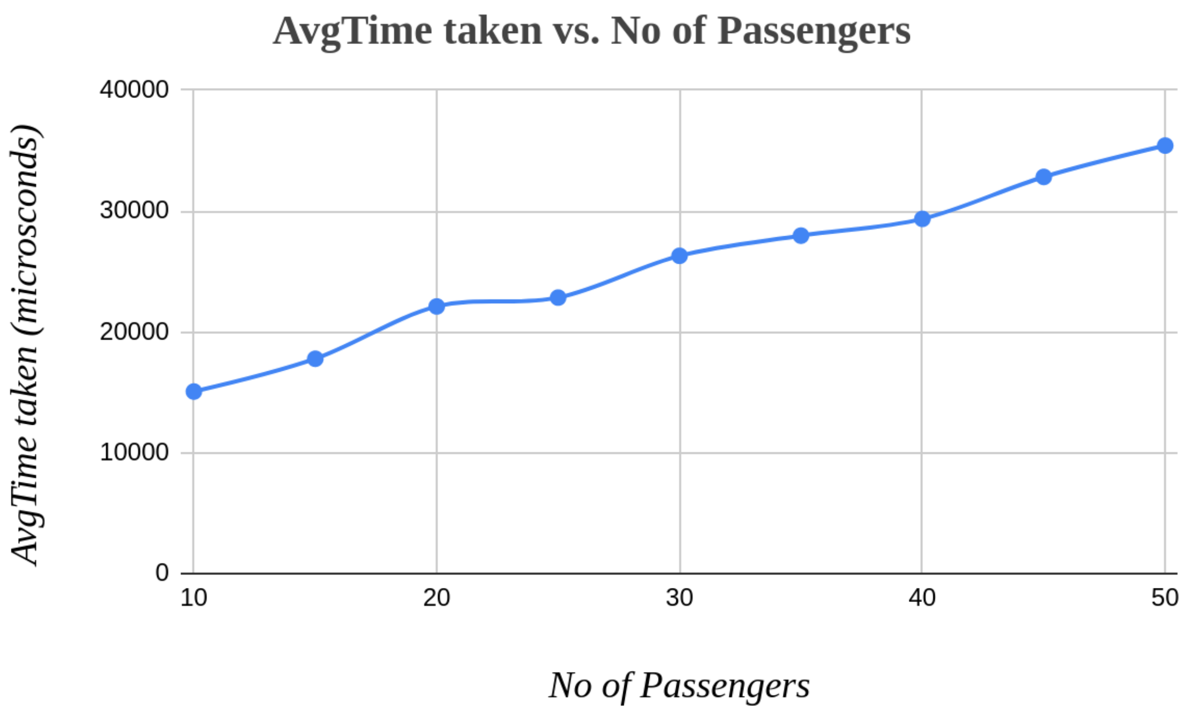
Destroy semaphore: -

```
// destroy semaphores
   sem_destroy(&binSem);
   sem_destroy(&carAvail);
   for (i = 0; i < P; i++) sem_destroy(&passenger_sem[i]);
```

Data and Graphs: -

(a)Average time by passenger to complete tour vs No of passengers (C = 25, k=5, P vary from 5 to 50)

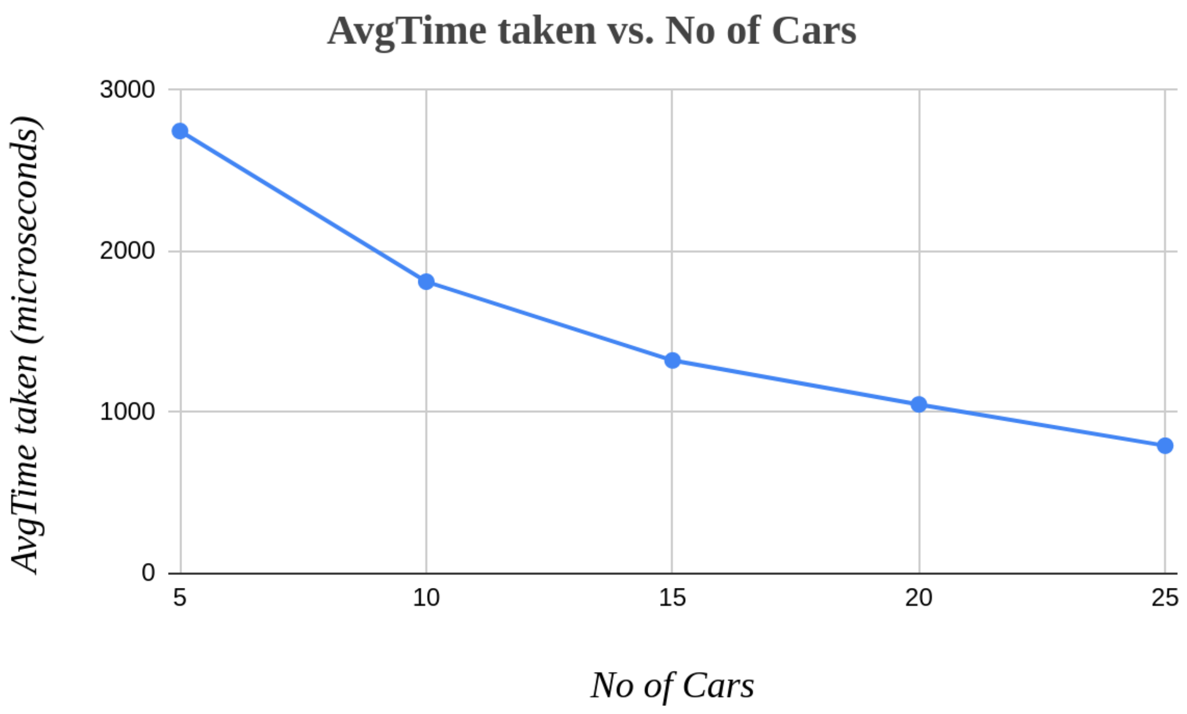
No of Passengers	**Time taken by passenger to completer tour (microseconds)**					**AvgTime taken**
	1st	**2nd**	**3rd**	**4th**	**5th**	
10	12786.3	18204.2	15901.2	14511.7	13908.7	15062.42
15	18626.1	18288.1	17354	16682.1	17932.8	17776.62
20	22010.7	23470.2	22178.3	23063.3	19805.9	22105.68
25	25057.9	20162.2	23122	22847.2	23002.7	22838.4
30	29335.8	24883.5	25739.3	19659.6	31869.7	26297.58
35	27197.8	26386.5	28307.2	29484.3	28459.1	27966.98
40	28609.9	30953.4	31127	27638	28443.2	29354.3
45	32190.7	35113	32121.9	35782	28933.7	32828.26
50	38083.1	33443	39624.2	29857.5	36103	35422.16



With an increase in the number of passengers, the average time taken by passengers to complete the tour increases, as more passengers are coming into the museum but no. of cars are constant.

(b) Average time taken by car to complete tour vs number of cars

No of Cars	Time taken by cars to completer tour (microseconds)					AvgTime taken
	1st	2nd	3rd	4th	5th	
5	2956	2542.6	2564.8	2841.4	2829.8	2746.92
10	1921.2	1750.6	1770.3	1881.2	1735.5	1811.76
15	1412.8	1312	1213.53	1570.87	1104.93	1322.826
20	1070.55	965.55	1104.7	966.25	1141.3	1049.67
25	1034.52	649.04	587.64	1007.56	688.48	793.448



With the increase in number of cars, average time taken to complete the tour is decreasing, this is because when more cars are introduced keeping passengers fixed more work in parallel is done.