Introduction

The solution presented in this report summarizes the algorithm needed to know how many distinct pairs of babies are ever at the crèche at the same time. The algorithm takes O(n^2) time complexity to compute the number of pairs of babies that are at the crèche at the same time.

Contact Tracing algorithmic solution

```
Main method{
    Scanner
    Print(Enter num of babies in creche)
    numBabies <- scanner.next.Int

    double[] arrival <- [numBabies]
    double[] depart <- [numBabies]

    for(i<-0 , I < numBabies,i++) {
        Print(Give the times of arrival and departure)
        arrival[i] <- scanner
        depart[j] <- scanner
    }
     count<-0
    for (int i = 0; i < numBabies; i++) {
      (int j = i + 1; j < numBabies; j++) {
         if ((arrival[i] >= arrival[j] && arrival[i] <= depart[j]) || (arrival[j] >= arrival[i]
&& arrival[j] <= depart[i]))
          count++;
Print(Number of distinct pairs of babies at the creche simultaneously:  + count)
```

Explanation of Algorithmic Solution

The scanner object reads input from the console. Print to the screen how many babies are in the crèche and store the input into numBabies as an int. The two arrays of type double with name ‘arrival’ and ‘departure’ stores the times of arrival and departure. The for loop starts from i=0 and stops at numBabies and prints out to the screen for the user to input both the arrival and departure times for n(numBabies) of times. This for loop has a Big-Oh O(n). Initialise a count variable of type int to see how many babies are at the crèche simultaneously. The nested for loop compares the arrival and departure timings of each pair of babies in the crèche to determine how many pairs of babies were present at the same time. The outer loop of the for loop iterates through each baby in the crèche, from i = 0 and ends with the last baby i = numBabies-1. The loop iterates n + 1 times and has a time complexity of O(n). The inner loop iterates through all the babies that come after the current baby in the outer loop, starting with the next baby (j = I + 1) and ending with the last baby (j = numBabies - 1). This loop iterates n + 1 times as well as n times because of there is an outer loop [n(n + 1)]. The if statement inside the inner loop of the for loop will check if the arrival time of one baby is between the arrival and departure times of the other baby. If this is true, this means that the two babies were at the crèche at the same time, and the count variable is incremented. This loop iterates n * n times because of the inner and outer for loops(n * n). The next line will print out the number of pairs of babies that are at the crèche at the same time.

The time complexiy of this program is:

n + 1 ---------- n + 1

[n(n + 1)] ---------- n^2 + n

n * n ---------- n^2

Adding all these together, we get 2n^2 + 2n + 1. We drop the contants and take the leading and this means that algorithm has a Big-Oh O(n^2).

Response to Bonus Question

No, we cannot improve on the time complexity of this algorithm because of the nested for loops. The nested for loops iterate to check if there are 2 babies simultaneously at the crèche. The outer loop of the for loop iterates through each baby in the crèche and the inner loop iterates through all the babies that come after the current baby in the outer loop. The if statement inside the inner loop of the for loop will check if the arrival time of one baby is between the arrival and departure times of the other baby. This means that the program has a time complexity of O(n^2). Without this step, the program won't be able to know how many babies are at the crèche.