Yuhao Zhuang

Homework 2

For the deque.h implementation, A unique pointer pointed to an array of type T is called at the initialization of the deque and front and tail integers are set to keep track of the first and last elements of the deque. PushBack is called to push more items at the end of deque and plus the tail index by one each, and if current tail is at the end while the front index is not 0 ,which means the array is not full yet, tail will wrap around the array and be set to 0. PushFront is called to push more items at the beginning of the deque and keeps reducing the front index by 1, and if current head position is at 0, which is the front of array, the head will wrap around the array, which sets the head index to the last element of array. If there is not enough space in either of these operations, the array will be resize with twice as many spaces, and the extra spaces will be added at the end of the array. PopFront is called to move elements in front of deque, and the front index will be increase by 1. If the front index is at the end of array, front will be set to 0, which is at the beginning of the array. PopBack is called to move elements in the back of deque, and the tail index will be reduced by 1. If the tail index is at the beginning of the array, tail will be wrapped around and set to the end of the array. To access certain elements, the deque will add the the index to the front index and check to see if it is within the size of the array, and if it is greater, the rest of the index will go to the front of the array and start counting from there. This implementation keeps moving the index of the front and back of the deque and when pushing and popping elements and it avoids the busy work of reindexing everything else in the array. Since the head and tail index can be wrapped either to the front or end of the array, this is a circular array implementation.This implementation access elements using the front and back index of the deque instead of returning the actual index of array.

For the plane_boarding, if the command arguments are correct, the deque will keep pushing back passengers in first class, which are smaller or equal to the second integer argument, and then it will keep pushing front passengers other than the first class to the same deque. The deque will then return the Back of the deque and call PopBack to remove it until the deque has no more passengers.

To test out if the deque was properly implemented, test cases were created to activate each individual if else statement inside each Push, Pop and other functions. For example, to test out the PushFront function when the front is at the beginning of the array, I call PushFront first, then PushBack, and finally PushFront again. I then call the get front and get back function and access the elements by the [] operator and compare them to the actual answers using google test to see if the function is properly implemented. I wrote as many test cases as I could think of to trigger every possible way the deque could be implemented to ensure the accuracy of this program.

Reference:
Piazza
Lecture Slides