CS261: Exam 1

INSTRUCTIONS

Rubric

1 Add Deque (65 points)

Complete the C function for adding element `val` to the front or back of Deque implemented as a dynamic array. Input arguments of the function include the pointer to Deque, element `val` , and `flag` that indicates addition to the front of Deque if `flag` = 1, and addition to the back if `flag` = -1. Your implementation should allow that indices of the front and end of Deque may "wrap around" the block of memory occupied by Deque.

2 Double Capacity Deque (35 points)

Complete the C function for increasing the memory capacity of Deque by 2. An input argument of the function is the pointer to Deque.

```
/* input: dq -- pointer to deque
          val -- value of the data element to be added
          flag -- flag = 1 => add to the front,
                  flag = -1 => add to the back
*/
void addDeque(struct Deque *dq, TYPE val, int flag)
{
/*5 points; check input arguments*/
    assert(dq && (flag == 1 || flag == -1));

/* 10 points; check memory capacity */
   if (dq->size == dq->capacity) _doubleCapDeque(dq);

/*10 points; check where to add*/
    if(flag == 1){  /* add to front */

/*10 points; Decrement the front index modulo capacity */
       dq->front--;
       if (dq->front < 0) dq->front += dq->capacity;
       /*alternative solution: dq->front = (dq->front - 1) % dq->capacity; */

/*5 points; add the new value*/
       dq->data[dq->front] = val;
   }
   else if (flag == -1){   /* add to back */

/*10 points; compute the back index modulo capacity*/
       int backIndex = (dq->front + dq->size) % dq->capacity;

/*5 points; add the new value*/
       d->data[backIndex] = val; /* add the new value */
    }
/*10 points; increment the size*/
    d->size ++;
}
```

```
/*-----------------------------------------------*/
/* Doubles the memory capacity of a deque */
void _doubleCapDeque (struct Deque *dq)
{
/*5 points; check input arguments*/
   assert (dq);
   int j;

/*5 points; memorize old data, old front index, old size, old capacity*/
   TYPE * oldData = dq->data;
   int oldFront = dq->front;
   int oldSize = dq->size;
   int oldCapacity = dq->capacity;

/*5 points; new memory allocation*/
   initDeque(dq, 2 * oldCapacity);

/*10 points; copy back old data*/
   for (j = 0 ; j < oldSize; j++) {
      dq->data[j] = oldData[oldFront++];
      if (oldFront >= oldCapacity) oldFront = 0; /*wrap around*/
   }

/*5 points; free memory*/
   free(oldData);

/*5 points; initialize new size to the old one*/
   dq->size = oldSize;
}
```