Nilesh Gupta

604489201

CS 31 Homework #5 Write Up

This assignment was a good exercise in pointers and helped solidify my understanding of the concept. The thing that really hit home here was the similarity between pointers and arrays.

The last three functions of the assignment took the most time. The reverse function involved swapping elements in the array equally far from the midpoint of the midpoint. Once that was determined, the rest of the function did not take much time.

The sort function took a little more time since the algorithm for sorting is not so obvious. It took a little time to realize that sorting the function could be done by swapping adjacent elements. However, the first approach I used was simply going through the array and swapping two elements if they were in non increasing order. It occurred to me that this would not create a universally sorted array. To fix this, I inserted another loop to get an element that was further along in the array to its rightful position.

The last function, sum\_31, took the most time. Initially, I was overwhelmed with all the possible combinations I would have to account for. Then, upon seeing some advice on piazza, I wrote down the problem on a piece of paper and the solution became much more apparent. For each 5 element array, I would have to account for the combinations between 2, 3, 4, elements and the one combination for all 5 elements. This made it a lot easier to conceptualize. The structure of the function just entailed the use of multi-nested loops and a separation between the arrays shorter than 5 and those 5 or longer.

The one drawback of my method for sum\_31 is the repetition of combinations the computer accounts for. For example, it considers (element1 + element2) and (element2 + element1) even though the expressions have the same value. One way to fix this might be instead of assigning all the loop variables to i, assign each loop variable after j to the one preceding it plus one to increase possible efficiency (e.g. m = l + 1, l = k + 1, k = j + 1, j = i).

(see next page for test cases)

| Test Input | Rationale |
| --- | --- |
| PHB\_LIST\_LENGTH = 1001 | This is outside of the max value for PHB length |
| PHB\_MAX\_VAL = 101 | This is outside of the max value for an element in PHB |
| bool PHB\_list\_check (in\_list, -1, out\_list) | The function will return false because in\_len < 0 |
| in\_list = {1, -1}  bool PHB\_list\_check (in\_list, in\_len, out\_list) | This function will return false since values in the input list are negative |
| in\_list = {1, -1}  bool PHB\_list\_delete\_duplicates (in\_list, in\_len, out\_list) | This function will return false since values in the input list are negative |
| in\_list = {1, -1}  bool PHB\_list\_fill\_up (in\_list, in\_len, out\_list) | This function will return false since values in the input list are negative |
| in\_list = {1, -1}  bool PHB\_list\_reverse (in\_list, in\_len, out\_list) | This function will return false since values in the input list are negative |
| in\_list = {1, -1}  bool PHB\_list\_sort (in\_list, in\_len, out\_list) | This function will return false since values in the input list are negative |
| in\_list = {1, -1}  bool PHB\_list\_sum\_31(in\_list, in\_len, out\_list) | This function will return false since values in the input list are negative |
| in\_list = {1, 101}  bool PHB\_list\_check(in\_list, in\_len, out\_list) | The function will return false since values in the input list are outside the 1-100 range |
| in\_list = {1, 101}  bool PHB\_list\_delete\_duplicates(in\_list, in\_len, out\_list) | The function will return false since values in the input list are outside the 1-100 range |
| in\_list = {1, 101}  bool PHB\_list\_fill\_up(in\_list, in\_len, out\_list) | The function will return false since values in the input list are outside the 1-100 range |
| in\_list = {1, 101}  bool PHB\_list\_reverse(in\_list, in\_len, out\_list) | The function will return false since values in the input list are outside the 1-100 range |
| in\_list = {1, 101}  bool PHB\_list\_sort(in\_list, in\_len, out\_list) | The function will return false since values in the input list are outside the 1-100 range |
| in\_list = {1, 99}  \*in\_len = 3;  bool PHB\_list\_sum\_31(in\_list, in\_len, out\_list) | The function will return false since values in the input list are outside the 1-100 range |
| in\_list = {1, 99}  \*in\_len = 3;  bool PHB\_list\_check(in\_list, in\_len, out\_list) | The program does not account for the user submitting inaccurate arguments (such as the length of the list being wrong) |
| in\_list = {1, 99}  \*in\_len = 3;  bool PHB\_list\_delete\_duplicates(in\_list, in\_len, out\_list) | The program does not account for the user submitting inaccurate arguments (such as the length of the list being wrong) |
| in\_list = {1, 99}  \*in\_len = 3;  bool PHB\_list\_fill\_up (in\_list, in\_len, out\_list) | The program does not account for the user submitting inaccurate arguments (such as the length of the list being wrong) |
| in\_list = {1, 99}  \*in\_len = 3;  bool PHB\_list\_reverse (in\_list, in\_len, out\_list) | The program does not account for the user submitting inaccurate arguments (such as the length of the list being wrong) |
| in\_list = {1, 99}  \*in\_len = 3;  bool PHB\_list\_sort (in\_list, in\_len, out\_list) | The program does not account for the user submitting inaccurate arguments (such as the length of the list being wrong) |
| in\_list = {1, 99}  \*in\_len = 3;  bool PHB\_list\_sum\_31 (in\_list, in\_len, out\_list) | The program does not account for the user submitting inaccurate arguments (such as the length of the list being wrong) |
| bool PHB\_list\_short(list, len, PHB\_LIST\_ORDER\_SAME) | The PHB\_LIST\_ORDER in the argument is not valid so this will cause the function to error |