Mansfield/Stoddard function list ASSN4

```
////////////////////////////////////////////////////////////////////////
///CODE FILENAME: MansfieldStoddard-ASSN4-SortProg.cpp
///DESCRIPTION:   Program tests the speed of four different sort techniques.
///  DATE:     26 APRIL 11
///  DESIGNER: Jason N Mansfield
///  FUNCTIONS:   driverFunction(), bubbleSort(), mergeSort(),quickS()
///               quickSort(), insertionSort(), exchange(), createArrays()
///               menuErrorCheck(), pickSorts(), doSort(), clockStop(),
///               verifyARRAY(), sortName()
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
///  FUNCTION: main()
///  DESCRIPTION:   simple calls driver function.
///  CALLS TO:  List of programmer-written functions called (names only)
///  IMPLEMENTED BY: Jason Mansfield
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
///  FUNCTION: driverfunction()
///  DESCRIPTION:  the driverfunction handles creating of lists followed by
///               cycling them through functions which verify user input,
///               select sort methods, perform selected sorts, timing of sorts,
///               creating new random lists and selection from user of times in which
///               to run.
///  CALLS TO: createArrays(), pickSorts(), soSort(), driverFunction()
///  IMPLEMENTED BY: Jason Mansfield
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
///  FUNCTION:  bubbleSort()
///  DESCRIPTION:  simple bubble sort function
///  INPUT:
///         Parameters: list[], int left, right.
///                random int in list[].
///  OUTPUT:
///         Return Val: list[]
///         Parameters: returns list[] sorted.
///  CALLS TO: exchange()
///  IMPLEMENTED BY: Jason Mansfield
////////////////////////////////////////////////////////////////////////
```

```
//////////////////////////////////////////////////////////////////////////
///  FUNCTION: exchange()
///  DESCRIPTION:  exchanges selected int's in array
///  INPUT:
///         Parameters: list[], int back, int front
///                front is an index in list as is back.
///  OUTPUT:
///         Return Val: *list
///         Parameters: list now has two int swapped.
///  IMPLEMENTED BY: Jason Mansfield
//////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////
///  FUNCTION: mergeSort()
///  DESCRIPTION:  merges and then sorts list
///  INPUT:
///         Parameters: list merge[] and int index selections left and right
///  OUTPUT:
///         Return Val: sorted list merge[]
///  CALLS TO:  mergeSort(), exchange()
///  IMPLEMENTED BY: Jason Mansfield
//////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////
///FUNCTION:    createArrays()
///DESCRIPTION: inserts random int into arrays
///INPUT:
///Parameters: random int make
///OUTPUT:
///Return Val: int *arrayOne and *arrayTwo filled with random int
///CALLS TO:  randMAKE
///IMPLEMENTED BY: Jason Mansfield
//////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////
///FUNCTION:    pickSorts()
///DESCRIPTION:  retrieve user input to chooce two sort functions
///INPUT:
///Parameters: string choice
///OUTPUT:
///Return Val: string choice
///CALLS TO:  menuErrorCheck, pickSorts
///IMPLEMENTED BY: Mansfield
//////////////////////////////////////////////////////////////////////////
```

```
//////////////////////////////////////////////////////////////////////////
///  FUNCTION: doSort()
///  DESCRIPTION:  Calls functions with function pointer and runs a variety
///                of sorts.
///  INPUT:
///         Parameters: unsorted list[], char sort (user selection)
///  OUTPUT:
///         Return Val: returns timer which contains the run time of sorting
///                method.
///         Parameters: double timer
///  CALLS TO:  mergeSort(), quickS(),insertionSort(), bubbleSort(), clockStop(),
///             verifyARRAY().
///  IMPLEMENTED BY: Jason Mansfield
//////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////
///FUNCTION:    menuErrorCheck()
///DESCRIPTION: verifies user input for mistakes
///INPUT:
///Parameters: string choice
///OUTPUT:
///Return Val: true of false if user submitted valid choice
///Parameters: bool check
///IMPLEMENTED BY: Jason Mansfield
//////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////
///FUNCTION:   clockStop()
///DESCRIPTION: displays time required to run sort
///Parameters: clock() stop time
///OUTPUT:
///Return Val: clock stop time minus start time giving total time of sort run
///CALLS TO: n/a
///IMPLEMENTED BY: Jason Mansfield
//////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////
///  FUNCTION: verifyARRAY()
///  DESCRIPTION:   verifies that indexes above the below indexes is larger throughout
///                 entire array; Otherwise a Warning message is given.
///  INPUT:
///         Parameters: sorted list[]
///  IMPLEMENTED BY: Jason Mansfield
//////////////////////////////////////////////////////////////////////////
```

```
////////////////////////////////////////////////////////////////////////
///  FUNCTION: sortName()
///  DESCRIPTION:  recieves user selected char and creates string named
///              after choosen sort method.
///  INPUT:
///         Parameters: char picked
///  OUTPUT:
///         Return Val: string name
///  IMPLEMENTED BY: Jason Mansfield
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
///  FUNCTION: quickS()
///  DESCRIPTION:  decrements length of array to be read from. This avoids
///              the quicksort from attempting to read past allocated memory.
///  INPUT:
///         Parameters: list array[] and indexes front and back.
///  OUTPUT:
///         Return Val: returns sorted array *array
///  CALLS TO:  quickSort()
///  IMPLEMENTED BY: Jason Mansfield
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
///  FUNCTION: quickSort()
///  DESCRIPTION:   quickSort breaks the list up via pivot points and partitions
///              the array for a divide and conquer approach.
///  INPUT:
///         Parameters: list index[] with current index in the front of
///              partition start while stop is the end of the partition.
///  OUTPUT:
///         Return Val: return sorted list index[] or recurse for furthur partitioning
///              and exchanges.
///  CALLS TO:  exchange(), quickSort()
///  IMPLEMENTED BY: Jason Mansfield
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
///  FUNCTION:  insertionSort()
///  DESCRIPTION:  a basic insertion sort algorithm which takes list and sorts
///              it from less than to greatest.
///  INPUT:
///         Parameters: int array list[] and index first and last.
///  OUTPUT:
///         Return Val: sorted int array list[]
///  IMPLEMENTED BY: Jason Mansfield
////////////////////////////////////////////////////////////////////////
```