Ryan Ellis

Lab report: In this lab, an algorithm for a double-ended selection sort was written to see if the timing complexity would be similar or better than the normal selection sorting algorithm. This lab took me approximately an hour to complete, cumulatively.

Pre-lab: Review of the design, implementation, and complexity of each of the following algorithms:

- Bubble sort
- Selection sort
- Insertion sort
- Merge sort
- Quicksort
- Radix sort

Lab 1.1

Exercise 1:

Source Code:

Header file

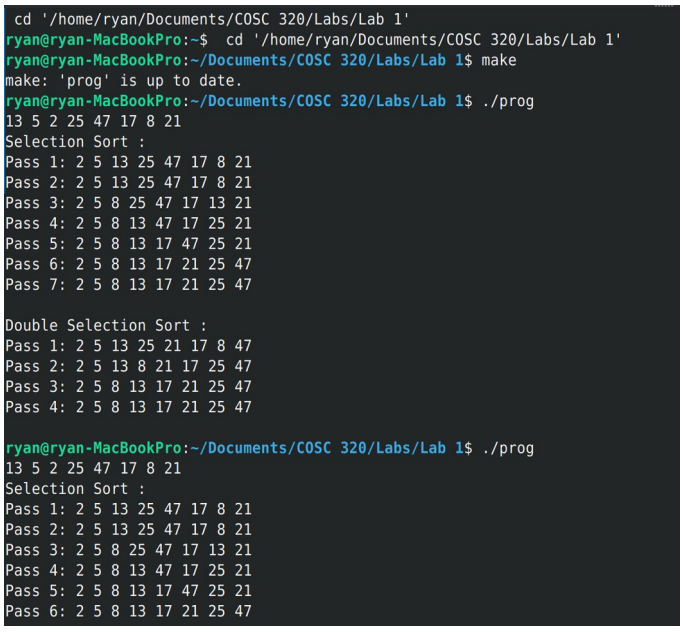
```
#ifndef DESELSORT_H
#define DESELSORT_H

#include <utility>
#include <vector>
#include <iostream>

using namespace std;

template <class T>
void deSelsort( T arr[], int size);
template <class T>
void printArray(T arr[], int size);
template <class T>
void selectionSort(T arr[], int size);

#endif
template <class T>
void printArray(T arr[], int size ){        //function to print array
   for(int i = 0; i < size; i++){
         cout<< arr[i]<< " ";
   }
   cout<<endl;
}
template <class T>
void selectionSort(T arr[], int size ){     //regular selection sort
   int minIndex, minValue;
```

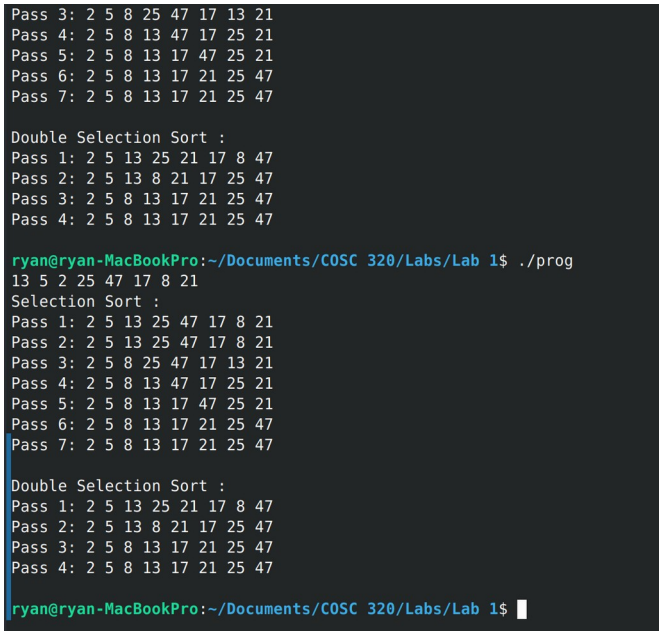
```
   for(int start = 0; start < (size - 1); start++){
      minIndex = start;
      minValue = arr[start];
      for(int index = start +1; index < size; index++){
         if (arr[index] <minValue){             //finds min value and swaps index
            minValue = arr[index];
            minIndex = index;
         }
      }

      swap(arr[minIndex], arr[start]);        //swap min value at index

      cout<< "Pass "<< start + 1<< ": ";      //print out array at pass
      printArray(arr, size );
   }
}

template <class T>
void deSelsort(T arr[], int size){          //double ended selection sort

   int minIndex, maxIndex;             // min and max index, right side set at end of array
   int right = size -1;

   for(int left = 0; left < right; left++, right--){       //for loop to end once left and right meet, increase left,
decrease right

      minIndex = left;
      maxIndex = right;          //set min to left, max to right

      for(int index = left; index <= right; index++){     // nested for loop with index set to left, less than or
equal to right

         if( arr[index] < arr[minIndex]){            //conditional statements to set index to min, max dependent
on greater/less than
            minIndex = index;
         }
         if(arr[index] > arr[maxIndex]){
            maxIndex = index;
         }
      }

      swap(arr[left], arr[minIndex]);         //swap values

      if(maxIndex == left){
         maxIndex = minIndex;           //set max index to min if it was at left before swap
      }

      swap(arr[right], arr[maxIndex]);

      cout<< "Pass " << left + 1<< ": ";      //print array at pass
      printArray(arr,size );

   }

}
```

Implementation (main program) :

```
#include "deSelsort.h"
#include <iostream>

using namespace std;

void div();     //prototype

int main(){

   int array[8] = {13, 5, 2, 25, 47, 17, 8 , 21};      //array declaration

   for(int i = 0; i < 8; i++)          //print original array
      cout<< array[i] << " ";
   div();
   cout<< "Selection Sort : "<<endl;   //regular selection sort function call
   selectionSort(array, 8 );
   div();
   int array2[8] = {13, 5, 2, 25, 47, 17, 8 , 21};

   cout<< "Double Selection Sort : "<<endl;        //double ended selection sort function call
   deSelsort(array2, 8);
   div();

   return 0;

}

void div(){
   cout<<endl;     //function to print line
}
```

Terminal Output:





[answers for exercise]

……

Lab Questions:

1) When does the sorting algorithm end?

The algorithm ends when both the indices for the inner and outer for loops reach the middle and all the comparisons have been made for each element within the array.

2) What is the time complexity of the double-ended selection sort? Is it better than the selection sort?

The timing complexity for the double-ended selection sort is still the same as the normal selection sorting algorithm ($O(n^2)$), as it uses the same structure of a nested for loop, so the comparisons are still being used in the same way, however with the double-ended selection sort we reach the end of a sorted array with less passes through the loops.