**CS311 Yoshii - HW2 Part 2 – Sorting Analysis and String Matching (based on Week4B and 5A)**

**DUE: Week 5 Sat**

**TOTAL: 40 points Your score is:**

**Your NAME: Salma Rashed**

**Date Turned in:2/26/2022**

**A) Review Questions [1pt per question = 5pts] Your score is:**

**Type your answers here. Answer completely.**

1. **What is the advantage of Insertion Sort over Selection Sort?**

**Insertion sort :**

**B(N)=N-1 comparisons which is fast**

**W(N)=N^2 comparisons which is slow**

**Selection Sort:**

**B(N)=W(N)=N^2 comparisons which means that in both cases it is slow**

1. **Rika-Chu Sort corrects one inversion per comparison.**
2. **What is its worst case number of comparisons?**

**W(N)= N^2 comparisons**

1. **Why?**

**It will be slow and there could be N(N-1)/2 bad pairs**

1. **What is the advantage of using Merge Sort over Quick Sort?**

**Merge Sort : B(N)=W(N)=N log (N) comparisons**

**Which means that it always has the same amount of recursion**

**Quick Sort: W(N)=N^2 , which is slow**

**B(N)=N log N ,**

**Which means that quick sort can have too many recursion or very few recursions**

**To conclude , merge sort is more time efficient**

1. **A) What is the disadvantage of using Merge Sort over Quick Sort?**

**Even though it is time efficient but it wastes space in combining list**

**b) Why is Radix sort unrelated to the F(n) = O(nlogn) theorem?**

**Radix sorts items by grouping in buckets , which means that it is not an algorithm based on comparisons .**

**5. Refer to the KMP String Matching table in the notes. If**

**matched p1=a p2=a p3=b p4=b p5=a p6=a mismatched p7=b**

S a a b b a a c …

P a a b b a a b …

P’ = **aabbaa**  f(6) = **2**

try p3 against ‘c’ Fill in the **???**s.

**B) Insertion Sort [1 per prompt=7pts] Your score:**

**Sort 231 123 324 20 23 56 (6 items)**

**using Insertion Sort and fill in the answers below to do just the first 2 steps!**

**Step 1 Put pos 2 (i.e. 123) in X index:**

1. **Which items were shifted to the right?**

**230**

1. **How many element-element comparisons until X is deposited back?**

**1**

**231>123 so 123 will become the first element**

1. **The resulting list is?**

**123 231 324 20 23 56**

**Step 2 Put pos 3 in X index:**

1. **Which items were shifted to the right?**

**No shifts, nothing is greater than 324**

1. **How many element-element comparisons until X is deposited back?**

**2 comparison**

**324>231**

**324>123**

1. **The resulting list is?**

**123 231 324 20 23 56 , same as before**

**No need to do the rest of the steps.**

**4. Re-arrange the above 6 elements so that you would have made the worst number of comparisons:**

**20 23 56 123 231 324**

**C) Radix Sort [1 per prompt=6pts] Your score:**

**Sort 231 123 324 100 230 560 (6 items)**

**using Radix Sort.**

**Hint: Use 0-list, 1-list, 3-list, 4-list etc. named after the char you are looking at.**

**Pass1:**

**Show the sub-lists here based on the last char**

**0 list->100 230 560 , all end with zero**

**1 list-> 231 , end with 1**

**3 list-> 123 , end with 3**

**4 list->324 , end with 4**

**Show the combined list**

**100 230 560 231 123 324**

**Pass2:**

**Show the sub-lists here based on the second char**

**0 list->100 , middle number is 0**

**2 list->123 323 , middle number is 2**

**3 list-> 230 231 , middle number is 3**

**6 list -> 560 , the middle number is 6**

**Show the combined list**

**100 123 323 230 231 560**

**Pass3:**

**Show the sub-lists here based on the first char**

**1 list-> 100 123**

**2 list -> 230 231**

**3 list-> 324**

**5 list -> 560**

**Show the combined list**

**100 123 230 231 324 560**

**D) Program: Quick Sort’s Partition [2+9=11 pts] Your score:**

**Code:**

**Comments:**

**Test results (required):**

**Total 9 points:**

**Q’s 2 points:**

**Q) State of the program [2pts]**

* **Does your program compile without errors? If not, describe:**
* **List any bugs you are aware of, or state “No bugs”:**
* **No ADT is needed. Just one source code file. – Run my solution program first!**
* **Using Notes-4B.doc, code only the Procedure Partition of Quick Sort.**
* **Must use partition.cpp**
* **int Partition** will take 1 array.
* It will partition the array in place into the small (< pivot) and large parts.
* It will always use the first element of the list as the pivot.
* Must use the algorithm that is in the partition.cpp file.
* Returns the first slot number of the Larger section (or 1 if there is no smaller section).
* **Hint:** while (left <= right) will help.

**Your main()**

1. Will declare an array of 10 slots.
2. Will ask the user to type integers into the array.
3. Then it will call Partition.
4. Display what is in the array with | separating the two sections.

**Required Test Cases: (Must test in this order)🡺Test1.txt**

1. **1 7 5 6 4 2 3 🡺 only 1 is in the left section (special case)**
2. **8 7 6 5 4 3 2 1 🡺 only 8 is in the right section**
3. **5 1 4 3 6 7 8 2 🡺 things less than 5 are in the left section**
4. **4 3 1 2 7 5 🡺 things less than 4 are in the left section**
5. **2 6 4 1 7 🡺 only 1 is in the left section**

**E) Program: Merge Sort’s Combine [2+9=11 pts] Your score:**

**Code:**

**Comments:**

**Test results (required):**

**Total 9 points:**

**Q’s 2 points:**

**Q) State of the program [2pts]**

* **Does your program compile without errors? If not, describe:**
* **List any bugs you are aware of, or state “No bugs”:**
* **No ADT is needed. Just one source code file. – Run my solution program first!**
* **Using Notes-4B.doc, code only the Procedure Combine of Merge Sort.**
* **Must use combine.cpp.**

**Void Combine** will take 3 **vectors** as arguments: A, B and R.

Combine should work for any size vectors as long as the size of A and B are the same.

It will combine the elements of A and B into R to produce the sorted list R.

You should know how to find the size of a vector.

Display “comparison” every time an element-element comparison is done.

**Your main()**

1. Will declare three vectors L1, L2 and L3.
2. Will ask the user to type integers in increasing order into L1.
3. Then ask the user to type more integers in increasing order into L2.
4. Then it will call void Combine to combine L1 and L2 to produce L3 which is passed back by reference.
5. Display what is in L3.

**Required Test Cases: (Must test in this order)🡺Test2.txt**

1. **Combine 1 2 3 with 4 5 6**
2. **Combine 1 3 5 with 2 4 6**
3. **Combine 4 5 6 with 1 2 3**
4. **Combine 1 2 5 6 with 3 4 7 8**

**Make sure the number of comparisons is what you expected.**

**\*\* You can try your completed functions in Merge.cpp and Quick.cpp in HW2demos if you want to \*\***

**SUBMIT THESE 5 FILES:**

1. **This assignment sheet with inserted answers.**
2. **Source code file (partitioncpp) of the program (with good comments).**
3. **Source code file (combine.cpp) of the program (with good comments).**
4. **Script (Test1.txt) of the compilation and test results.**
5. **Script (Test2.txt) of the compilation and test results.**

* **Whether working or not, test result must include the lines for compiling your files or we will not grade your program i.e. 0 points for the program.**
* **Did you check your comments and style against CS311 How To Comment.doc??**
* **Did you answer all the questions?**