DATA STRUCTURES

ASSIGNMENT 2

Submitted by

Apnatva Singh Rawat

21354929

Task 2:

 Pivot: First Element - I tried to use a median pivot but I was unsure of my program and decided to use a simpler approach. I successfully implemented Quick Sort using the first and last element as pivot separately.

 Partition Scheme: Hoarse Partition: This was discussed in class and I quickly caught on to it so I successfully implemented it for this assignment.

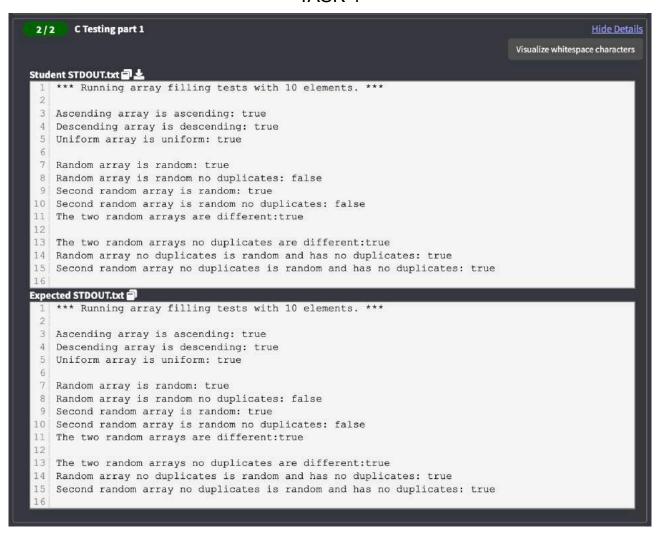
Task 3:

- The number of comparisons and swaps were maximum for selection sort, proving it has a similar run time for all cases.
- The number of swaps for correctly sorted / uniform array for insertion sort were 0, as I expected them to be.
- For random arrays it did not work well and thus resulted in very high number of comparisons and swaps as was expected.
- For quick sort I used the first element as pivot. For sorted arrays it would give a better run time if I chose the median as the pivot. Worked similar for first and last element as pivot. Performed better when sorting the two random arrays

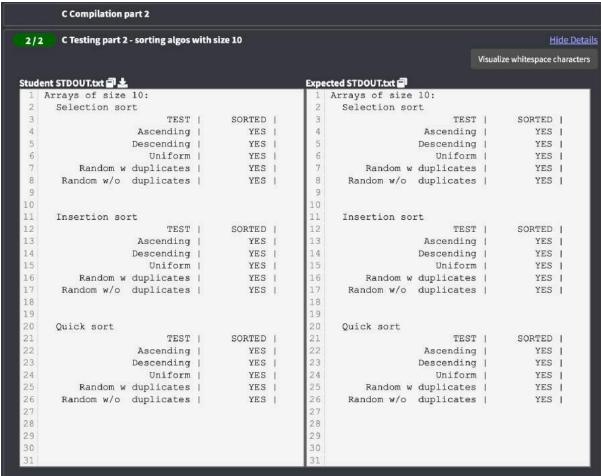
Task 4:

- I successfully wrote and executed the program although the output slightly differed in values and a difference in spacing of the output
- To print the top 5 games of each year
 - Load all values from the CSV file
 - Sort the data by release date.
 - Now all elements of that year will be grouped (ascending / descending).
 - Find groups and keep account of the first and last element of each group
 - Sort each group individually by score received (descending).
 - The first 5 elements of each group will be the highest rated.
 - Again look for groups, print first 5 elements of each group

TASK 1

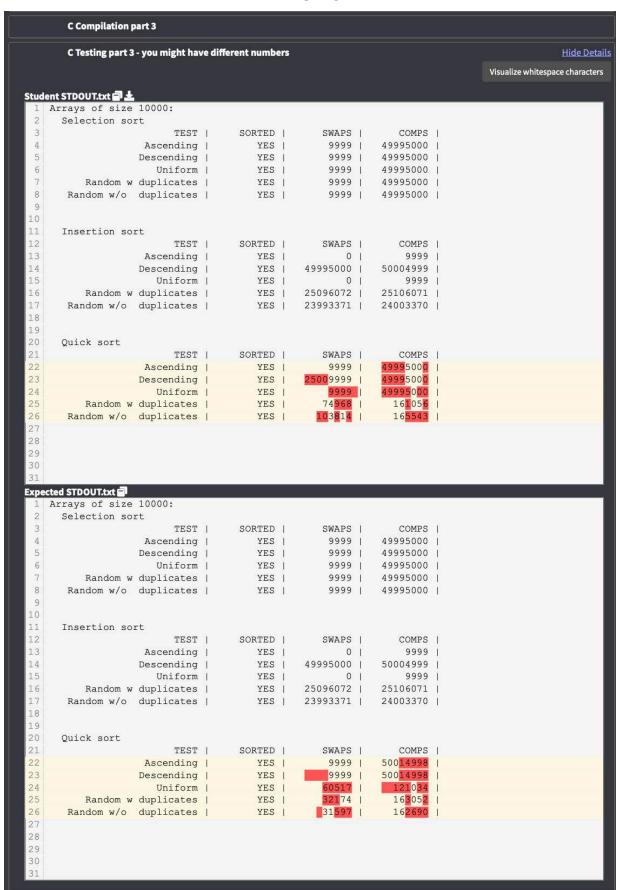


TASK 2



esta esta esta esta esta esta esta esta	/ 3 C Testing part 2 - sorting algos with size 10 000									Hide Det		
									Visuali	ize whitespace o	haracte	
Student STDOUT.txt 🗗 🕹							Expected STDOUT.txt 🗐					
1	Arrays of size	10000:				1	Arrays of size	10000:				
2	Selection sort					2	Selection sor	t				
3		TEST	1	SORTED	1	3		TEST	1	SORTED	1	
4		Ascending	1	YES	1	4		Ascending	1	YES	1	
5		Descending	1	YES	F	5		Descending	1	YES	1	
6		Uniform	1	YES	1	6		Uniform	- 1	YES	1	
7	Random w	duplicates	1	YES	1	7.	Random w	duplicates	1	YES	1	
8	Random w/o	duplicates	1	YES	1	8	Random w/o	duplicates	1	YES	1	
9						9						
1.0						10						
11	Insertion sort					11	Insertion sor	t				
12		TEST	1	SORTED	F	12		TEST	1	SORTED	1	
13		Ascending	1	YES	1	13		Ascending	1	YES	1	
14		Descending	1	YES	1	14		Descending	1	YES	1	
15		Uniform	1	YES	1	15		Uniform	. 1	YES	1	
16	Random w	duplicates	1	YES	1	16	Random w	duplicates	1	YES	1	
17	Random w/o	duplicates	1	YES	1	17	Random w/o	duplicates	1	YES	1	
18						18						
19						19						
20	Quick sort					20	Quick sort					
21		TEST	1	SORTED	1	21		TEST	1	SORTED	1	
22		Ascending	1	YES	1.	22		Ascending	1	YES	1	
23		Descending	1	YES	1	23		Descending	1	YES	1	
24		Uniform	1	YES	1	24		Uniform	1	YES	1	
25	Random w	duplicates	1	YES	Ĭ.	25	Random w	duplicates	1	YES	1	
26	Random w/o	duplicates	1	YES	1	26	Random w/o	duplicates	1	YES	1	
27						27						
28						28						
29						29						
30						30						
31						31						

TASK 3



TASK 4

