**Sorting Algorithms**

**3A. Binary Search:** Implement a program to search and find the student details in

an efficient manner. Reduce the number of comparisons as much as possible. Use the student’s registration number as the key. Store the student’s name, registration number, phone number and CGPA in the list of student details.

**Algorithm:**

Step 1: …

Step 2: …

Step 3: …

Step 4: …

Step 5: …

Step 6: …

Step 7: ...

**Program:**

**Output:**

**Results:**

Thus, the program to search and find student details in an efficient manner is implemented.

**3B. Application of Binary Search:** Implement a program to find the square root of a number. User can give the number randomly. Floor the result in case of floating point.

**Algorithm:**

Step 1: …

Step 2: …

Step 3: …

Step 4: …

Step 5: …

Step 6: …

Step 7: ...

**Program:**

**Output:**

**Results:**

Thus, the program to find the square root of a number using binary search is implemented.

**2C. Bubble Sort:** Sort the given array of elements in ascending order and print out the number of comparisons performed.

**Algorithm:**

Step 1: …

Step 2: …

Step 3: …

Step 4: …

Step 5: …

Step 6: …

Step 7: …

**Program:**

**Output:**

**Results:**

Thus, the program to sort the given array using bubble sort is implemented.

**2D.** **Insertion Sort:** Sort the given array of elements in ascending order and print out the number of comparisons performed.

**Algorithm:**

Step 1: …

Step 2: …

Step 3: …

Step 4: …

Step 5: …

Step 6: …

Step 7: ...

**Program:**

**Output:**

**Results:**

Thus, the program to sort the given array using insertion sort is implemented.

**2E. Selection Sort**: Sort the given array of elements in ascending order and print out the number of comparisons performed.

**Algorithm:**

Step 1: …

Step 2: …

Step 3: …

Step 4: …

Step 5: …

Step 6: …

Step 7: …

**Program:**

**Output:**

**Results:**

Thus, the program to sort the given array using selection sort is implemented.

**2F. Quick Sort**: Sort the given array of elements in ascending order and print out the number of comparisons performed.

**Algorithm:**

Step 1: …

Step 2: …

Step 3: …

Step 4: …

Step 5: …

Step 6: …

Step 7: ...

**Program:**

**Output:**

**Results:**

Thus, the program to sort the given array using quick sort is implemented.

**2G. Merge Sort**: Sort the given array of elements in ascending order and print out the number of comparisons performed.

**Algorithm:**

Step 1: …

Step 2: …

Step 3: …

Step 4: …

Step 5: …

Step 6: …

Step 7: …

**Program:**

**Output:**

**Results:**

Thus, the program to sort the given array using merge sort is implemented.

**2H.** Compare the number of comparisons of various sorting algorithms mentioned in above questions 3 to 6. Print a table which shows the input array, and number of comparisons performed by various algorithms. Reuse above sorting programs as functions in this new program.

**Algorithm:**

Step 1: …

Step 2: …

Step 3: …

Step 4: …

Step 5: …

Step 6: …

Step 7: ...

**Program:**

**Output:**

**Results:**

Thus, the Josephus problem is solved using a circular linked list.