**Assignment 1**

**Question 1**:Coin Toss Simulation

Through a coin toss simulation, show that probability of getting HEAD, by tossing a fair coin, is about 0.5. Write your observation from the simulation run.

**Solution**:

**Algorithm** CoinToss()

*//Function to simulate coin toss*

**{**

  outcome := rand(1);

  outcome := round(outcome);

**return** outcome;

**}**

**Question 2**: Sort Analysis

Implement two different versions of bubble sort for a randomized data sequence.

**Solution:**

**Algorithm** BubbleSort(arr)

*//Sorting an array using classical bubble sort*

**{**

   len = length(arr)

**for** i := 1 **to** len **do**

**{**

**for** j := i **to** len-1 **do**

**{**

**if** (arr[j] > arr[j+1]) **then**

**{**

           temp := arr[j];

           arr[j] := arr[j+1];

           arr[j+1] := temp;

**}**

**}**

**}**

**}**

**Algorithm** modifiedBubbleSort(arr)

*//Sorting an array using modified bubble sort.*

*//A* *flag is used to determine if array is already sorted*

**{**

 len := length(arr);

    sorted := true;

**for** i := 1 **to** len **do**

**{**

      sorted := true;

**for** j := i **to** len-1 **do**

**{**

**if** (arr[j] > arr[j+1]) **then**

**{**

            temp := arr[j];

            arr[j] := arr[j+1];

            arr[j+1] := temp;

*//If a swap occurs then array is not sorted.*

            sorted = false;

**}**

**}**

**if** (sorted) **then**

**return**;

**}**

**}**

Question 3: Determine the asymptotic behaviour of at least 5 internal sorting algorithms for each of the following input variations:

1. Random data
2. Reverse Ordered Data
3. Almost Sorted Data
4. Highly Repetitive Data

Select a suitable number of elements for the analysis that supports your program.

Solution:

**Algorithm** modifiedBubbleSort(arr)

*//Sorting an array using modified bubble sort.*

*//A* *flag is used to determine if array is already sorted*

**{**

 len := length(arr);

    sorted := true;

**for** i := 1 **to** len **do**

**{**

      sorted = true;

**for** j := i **to** len-1 **do**

**{**

**if** (arr[j] > arr[j+1]) **then**

**{**

            temp := arr[j];

            arr[j] := arr[j+1];

            arr[j+1] := temp;

*//If a swap occurs then array is not sorted.*

            sorted := false;

**}**

**}**

**if** (sorted) **then**

**return**;

**}**

**}**

**Algorithm** selectionSort(arr)

*//Sorting an array using selection sort.*

**{**

len := length(arr);

**for** i := 1 **to** len-1 **do**

**{**

      sorted := true;

**for** j := i+1 **to** len **do**

**{**

**if** (arr[j] > arr[i]) **then**

**{**

            temp := arr[j];

            arr[j] := arr[i];

            arr[i] := temp;

**}**

**}**

**}**

**}**

**Algorithm** heapSort(arr)

*//Sorting an array using modified bubble sort.*

**{**

 len := length(arr);

**for** i := 1 **to** len **do**

**{**

      sorted := true;

**for** j := i **to** len-1 **do**

**{**

**if** (arr[j] > arr[j+1]) **then**

**{**

            temp := arr[i];

            arr[i] := arr[j];

            arr[j] := temp;

**}**

**}**

**if** (sorted )

**return**;

**}**

**}**