Bahria University
Karachi Campus

Bahria University
Discovering Knowledge

COURSE: DATA STRUCTURES AND ALGORITHMS
TERM: SPRING 2020, CLASS: BCE- 4(A)

Submitted By:

NAME: **QASIM HASSAN** REG NO: **57485**

Assignment: 01

Submitted To:

Engr. Sidra Mudassar

Signed **Remarks:** **Score**_____

Question no 1:

List down the best, worst, average time complexity of all the data structures.

Answer:

Data Structures	**Best Time Complexity**	**Average Time Complexity**	**Worst Time Complexity**
Quick Sort	Ω(nlog(n))	Θ(nlog(n))	O(n^2)
Merge Sort	Ω(nlog(n))	Θ(nlog(n))	O(nlog(n))
Tim Sort	Ω(n)	Θ(nlog(n))	O(nlog(n))
Heap Sort	Ω(nlog(n))	Θ(nlog(n))	O(nlog(n))
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)
Tree Sort	Ω(nlog(n))	Θ(nlog(n))	O(n^2)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)
Shell Sort	Ω(nlog(n))	Θ(logn^2)	O(nlog(n^2))
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)
Cube Sort	Ω(n)	Θ(nlog(n))	O(log(n))
Stack	N/A	Θ(n)	O(n)
Queue	N/A	Θ(n)	O(n)
Single linked list	N/A	Θ(n)	O(n)
Double linked list	N/A	Θ(n)	O(n)
B-Tree	N/A	Θ(log(n))	O(log(n))
Binary Search Tree	N/A	Θ(log(n))	O(n)
AVL Tree	N/A	Θ(log(n))	O(log(n))
KD Tree	N/A	Θ(log(n))	O(n)
Splay Tree	N/A	Θ(log(n))	O(log(n))
Red-Black Tree	N/A	Θ(log(n))	O(log(n))
Cartesian Tree Hash Table	N/A N/A	Θ(log(n)) Θ(1)	O(n) O(n)
Skip List	N/A	Θ(log(n))	O(n)
Array	N/A	Θ(n)	O(n)

Question no 2:

Define an algorithm to reverse the string using an array.

Answer:

<u>Algorithm to reverse a String</u>:

1. Initialize or declare a string.
2. Define its length using length() function.
3. Repeat for string.length()-1 to 0
   [End of loop].
4. Exit.

Question no 3:

Define an algorithm that prints out all the subset of four elements of a set of n elements the elements of this set are sorted in a list that is input to the algorithm.

Answer:

<u>Algorithm to print all the subsets of four elements</u>:

1. Initialize an array of 4 (n) elements.
2. Repeat until all the elements are taken from the user
   [End of 1st loop].
3. Run three loops (nested loop) for the output of subsets as there are four elements so one will be removed at a time.
4. Repeat for i=0 to n-2 (for first element).
5. Repeat for j=i+1to n-1 (for 2nd element).
6. Repeat for k=j+1 to n (for 3rd element)
7. Print all the three elements
   [End of 2nd, 3rd and 4th loops]
8. Exit.

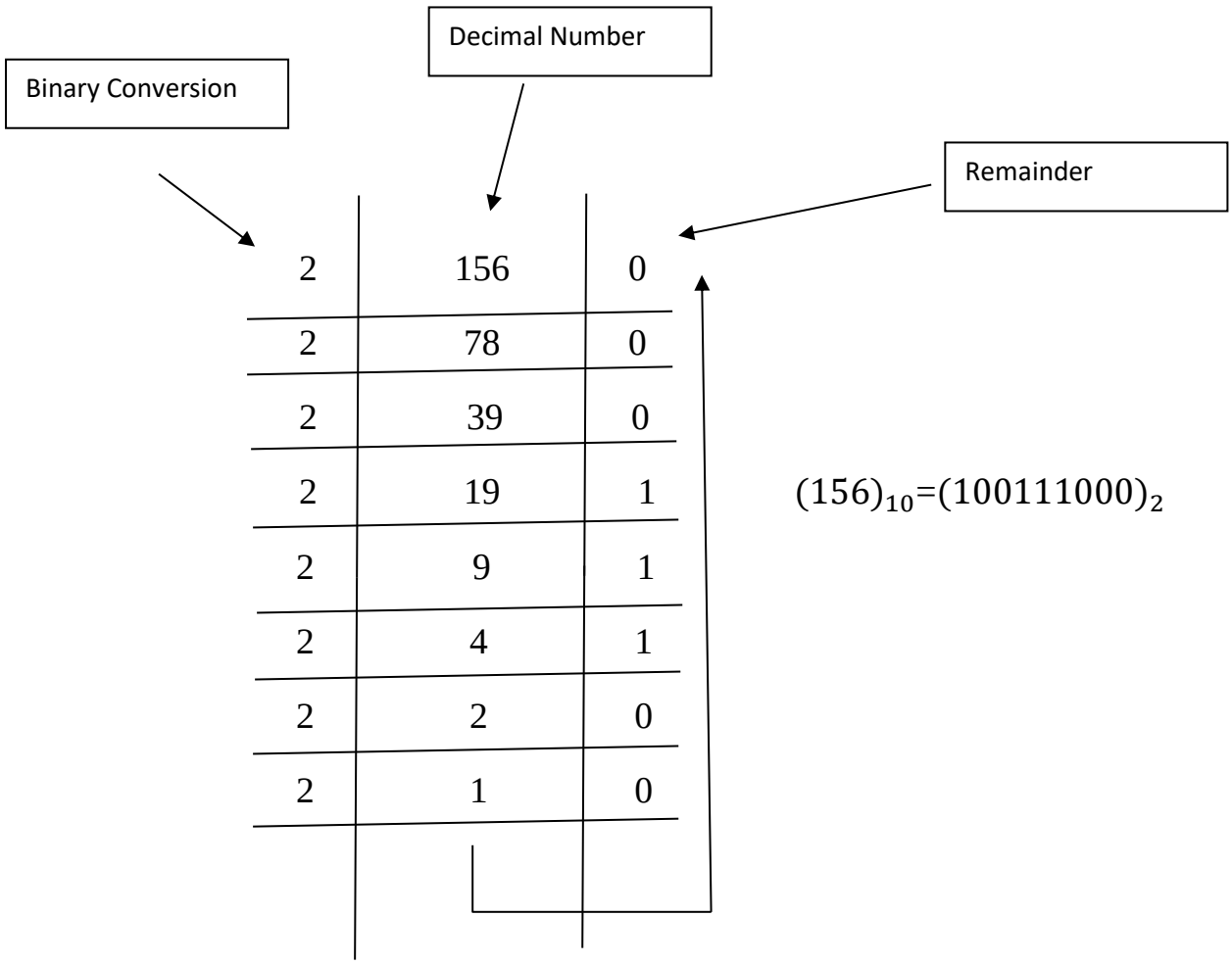
Question no 4:

A positive integer is input, Define a function (function name: binary) to find the binary equivalent of this number using recursion. For example, if input is 156, then binary value is 10011100 (no code is needed, just a sketch and pseudo code).

Answer:

Sketch:

Sketch the binary equivalent of 156 (Convert decimal into binary).



$(156)_{10}=(100111000)_2$

Pseudo Code:

1. Construct a function named binary and initialize a variable ‘rem’ (remainder) and declare ‘binarynum’ 0 and ‘temp’ 1.
2. Repeat while ‘decimalnum’ is not equal to 0.
3. Implement the following formulas until the condition is valid:
   rem=decimalnum%2
   decimalnum=decimalnum/2
   binarynum=binarynum+rem*temp
   temp=temp*10
   [End of loop].
4. Return binarynum.
5. In main() function, initialize ‘decimalnum’ and take it from user.
6. Call the binary() function.
7. Exit.