2. What is an Algorithm? Give five examples

Ans: **Algorithm** is a finite, deterministic and effective problem solving method used for solving complex and simple real time problem. It includes a set of instructions that helps in solving problems.

Examples: Adding an element to an array, Deleting data from list, Searching an element in a list, reverse an ordered list, Sorting, Binary Search etc

3. What is time and space complexity of an algorithm?

Ans: **Time Complexity** : The amount of time taken by an algorithm to run till the program is executed completely. There are three kinds of Time complexity.

1. Worst case(O)
2. Best case(omega)
3. Average case(theta)

**Space Complexity** : The amount of memory required by the algorithm to run is called as space complexity.

4. What is the time complexity of the following code, and why?

public makeSentence ( String[] words) {

String sentence=“”;

for (String w:words) {

sentence+=w;

}

return sentence;

}

Ans: **O(n^2)** because the string is Immutable type and the time complexity for copying is N. Also for iterating through the loop, will take at least n times. So its O(n^2).

5. What are all Stack operations, explain.

Ans: ISEMPTY, ISFULL, PUSH, POP, SIZE

1. ISEMPTY: If the stack is empty, it returns boolean value true. Else returns false.
2. ISFULL: It checks whether the stack is full then returns boolean value as true or false.
3. PUSH: Pushes the elements into the stack if the stack is empty or not full.
4. POP: It is used to remove the element pointed by the pointer of the stack.
5. SIZE: It returns the size of a stack.

7. Consider the following Node data structure, build a Stack linkedList with the following data:

{31, “Name1”}, {24, “Name2”}, {10, “Name3”}, {44, “Name4”}, {81, “Name5”},

{33, “Name6”}

a) Write java implementation for all necessary Stack operations using LinkedList.

b) Compile and run your program.

c) What is Stack linkedList time and space complexity?

class Node {

int Age;

String Name;

Node next;

}

Ans: **Time Complexity** ofpush, pop, isEmpty: O(1) and Time Complexity of print: O(n) => O(n)

**Space Complexity**: O(n)

8. Consider data: {31, “Name1”}, {24, “Name2”}, {10, “Name3”}, {44, “Name4”}, {81, “Name5”}.

a) Write Java code for Array implementation of Stack.

b) Compile and run the code.

c) What is time and space complexity of Stack Array implementation?

Ans: **Time Complexity** ofpush, pop, isEmpty: O(1) and Time Complexity of print: O(n) => O(n)

**Space Complexity**: O(n).

9. Suppose in problem-8 above, the array size was: a) too large, or b) too small. How would you

manage resizing the array for (a) and (b). Write the code, compile and test the program. Discuss

the running time/space complexity of your approach.

Ans: **Time complexity**: O(n) The time complexity of **memoryadd()** is O(n) + O(n) => O(n) and for **memorysub()** is O(n/2) => O(n).

**Space Complexity**: O(n).