1. (2 marks) make your java implementation of Fibonacci function using dynamic programming technique, submit your java file.

Your main function should invoke fibo(3), fibo(7), fibo(5), and paste your output below

|  |
| --- |
| A screenshot of a computer  Description automatically generated |

1. (4 marks) List out data structures you learnt in this course, then explain each in detail.

|  |
| --- |
| Data Structures:  Arrays, LinkedList, Stacks, Queues, Trees, Graphs, HashTables/Maps  Arrays - A continuous block of memory used to store elements of the same data type is known as an array. Using their index, it enables constant-time access to certain elements. Although arrays have set widths and offer quick access to and updates of its elements, expanding an array might be inconvenient.  LinkedList - Each node in a linked list is made up of data and a pointer or reference to the node that comes after it in the chain. It is a sort of linear data structure. Single links, which point only to the next node, and double links, which point to both the next and preceding nodes, are the two different forms of linked lists. They have slower access speeds than arrays but allow for efficient insertion and deletion.  Stacks - A stack, a form of Last-In-First-Out (LIFO) data structure, allows for additions and subtraction from the "top" end. The principle of "last in, first out" states that the last ingredient in will be the first one removed. The operations push and pop, which add and delete pieces from stacks accordingly, are commonly employed.  Queues - First-In-First-Out (FIFO) data structures like queues allow for the addition of elements at the back and the removal of elements at the front. It adheres to the tenet of "first in, first out," which states that the first piece introduced will also be the first withdrawn.  Trees - With a root node and child nodes arranged in a hierarchical manner, trees are hierarchical data structures. There may be 0 or more child nodes for each node. Binary trees each node has a maximum of two children, binary search trees left child is smaller, right child is larger, and balanced trees like AVL trees or red-black trees are examples of common types of trees.  Graphs - A collection of nodes vertices connected by edges make up a graph. Graphs can be undirected or directed where the edges all point in the same direction. They are applied in a variety of contexts, including network modelling, social networks, and more, to depict complicated relationships.  HashTables/Maps - Hash tables use a hash algorithm to associate keys with particular array indices, making retrieval, insertion, and deletion operations quick and simple. For key-value pairs, they offer constant-time average-case access. |

1. (4 marks) Describe listed algorithm design strategies: Brute Force, Decrease and Conquer, Divide and Conquer, Transform and Conquer, Greedy Technique, Graph Algorithm, dynamic programming.

|  |
| --- |
| Brute Force - Brute Force is a simple algorithmic strategy in which all potential solutions are systematically investigated to identify the best one. The best result is chosen after attempting every conceivable combination or permutation of inputs. Brute Force, while simple to grasp, can be extremely inefficient for large problems since it can cause the number of computations needed to grow exponentially.  Decrease and Conquer - In this method, an issue is divided into smaller, more manageable subproblems, each of which is solved before the larger problem is further simplified. Till the initial issue is resolved, this process is repeated. The "decrease" stage shrinks the problem to make it more manageable, while the "conquer" step resolves the smaller issue.  Divide and Conquer - Recursively breaking down a complicated problem into smaller, independent subproblems of the same kind is known as "Divide and Conquer." The final answer to the main problem is then reached by separately resolving the subproblems and combining the results. Divide and Conquer is most effective when the subproblems are much less than the main problem and can be solved on their own.  Transform and Conquer - This method simplifies the original problem or makes it simpler to address by changing how the problem is represented. The solution is then translated back to the original problem domain when the modified problem has been resolved. Transforming a problem into a different data format or representation is a common step in the Transform and Conquer strategy. This can result in algorithms that are more effective.  Greedy Technique - Making locally optimal decisions at each stage of the Greedy Technique is done in the hopes of locating a global optimum. It takes the best option currently accessible without taking into account the effects of future decisions, which is known as the "greedy" criterion. The Greedy Technique can be effective and deliver adequate results in many situations, even though it may not always produce the best answer for a given issue.  Graph Algorithm - Problems relating to graphs, which are mathematical structures made up of nodes vertices and edges connections between nodes, are the subject of graph algorithms. Numerous graph-related problems are addressed by these algorithms, including identifying the shortest path between two nodes, figuring out whether a graph is connected, locating cycles, and other graph traversal issues.  Dynamic Programming - Problems are broken down into overlapping subproblems using the dynamic programming technique, and each subproblem is only ever solved once. Subproblem solutions are kept in a table, and when a subproblem needs to be solved again, the precomputed solution is used rather than having to be computed from scratch. Dynamic programming is especially helpful for solving optimisation issues and is built to effectively deal with issues involving optimal substructure and overlapping subproblems. It aids in the avoidance of unnecessary computations and can greatly enhance the performance of some algorithms. |