1. What are the two main styles of parallelism? Explain.
    a. Instruction-level parallelism: Instruction-level parallelism (ILP) is the parallel or simultaneous execution of a sequence of instructions in a computer program. These instructions can be re-requested and assembled which are later executed simultaneously without influencing the after effect of the program.
    b. Task Parallelism: Task parallelism (also known as function parallelism and control parallelism) is a form of parallelization of computer code across multiple processors in parallel computing environments. Task parallelism focuses on distributing tasks—concurrently performed by processes or threads—across different processors. In contrast to data parallelism which involves running the same task on different components of data, task parallelism is distinguished by running many different tasks at the same time on the same data. It divides the tasks into subtasks, then allocates each of task for execution and finally processors concurrently perform subtask execution.
2. What are the two main types of localities? Explain.
    a. Temporal Locality: The improvement in performance resulting from the presence of the cache assumes that there is repeated reference to the same data item. This notion of repeated reference to a data item in a small-time window is called temporal locality of reference.
    b. Spatial Locality: Increased bandwidth results in higher peak computation rates. The data layouts were assumed to be such that consecutive data words in memory were used by successive instructions. In other words, if we take a computation-centric view, there is a spatial locality of memory access.
3. What are the three basic programming paradigms for parallel processing? Explain.
    a. There are three basic programming paradigms for parallel processing they are Explicit parallelism, Implicit parallelism and Ideal system. Explained as follows.
    b. Explicit Parallelism: It is a feature of programming language for parallel processing system which makes the programmer to write his program to differentiate which parts should be executed as independent parallel tasks. It gives high performance but more work to the programmer to indicate independent parallel tasks.

- c. Implicit Parallelism: It is a feature of a programming language for a parallel processing system which automatically decides which tasks should run in parallel. Low performance compared to explicit parallelism.
- d. Ideal System: The Ideal system is a situation in parallelism which allows to use both explicit and implicit parallelism techniques.

4. Discuss the difference between shared address space machines and distributed address space machines. Discuss the advantages and disadvantages of both architectures.
    - a. Shared Address Space: In this shared address space processors will be directly accessing all the data in the system. It naturally fits the shared address
        - i. Advantages:
            1. It is typically time shared.
            2. Access to job queue can be centralized or decentralized.
            3. Here in shared memory, it offers a unified address space in which all data can be found.
        - ii. Disadvantages:
            1. Access to data is delayed.
            2. Primary disadvantage is the lack of scalability between the memories and CPU's.
            3. It becomes increasingly difficult and expensive to design and produce shared memory machines with ever increasing number of processors
    - b. Distributed Address Space: In this Distributed address space each processor has a private memory, and nothing is shared between them they can only directly access the local data. It naturally fits the private address
        - i. Advantages:
            1. Simple to develop.
            2. Address space is easily expandable.
            3. Any CPU failure doesn't affect the whole system, it is highly reliable.
            4. Nodes of parallel machine have simple version of OS.

        ii. Disadvantages:
            1. It is difficult to program for distributed address space
            2. Usually, the access to parallel machine is via host computer running a serial OS.
            3. Computational tasks can only operate on local data

5.
    a. Parallel I/O is a subset of parallel computing that performs multiple input/output operations simultaneously, rather than processing I/O requests serially one by one it accesses data on disk simultaneously. We need parallel I/O because this allows a system to achieve higher write speeds and it maximizes bandwidth based on the principle that larger issues can be divided into multiple, smaller issues that can be solved at the same time. Used mostly in high-performance computing, parallelism can help run application efficiently and quickly.

6. Give two examples of anti-dependence and give the corresponding solutions to remove the dependence.
    a. Anti-dependence: A variable used in a statement is assigned to in a subsequent statement.
        i. Eg 1:
            1. Y= Z * Z
            2. Z=9
        ii. Eg 2:
            1. Z = 5
            2. X = Z + 1
            3. Z = 9
    b. An anti-dependency is an example of a name dependency. That is, renaming of variables could remove the dependency, as in the next example:
        i. Eg 3:
            1. Z = 3
            2. Z2 =Z
            3. X = Z2 + 1
            4. Z = 9

c. A new variable Z2 has been declared and value of Z is copied to it. Now we can execute these instructions in parallel. However, this modification has given new dependency, now instruction 2 is now truly dependent on instruction N and instruction N is truly dependent on instruction 1. Due to this flow dependencies these new dependencies are impossible to remove safely.

7.

a. If a sequential search of the tree is performed using the standard depth-first search (DFS), 11 edges need to be traversed to access the dark node. If traversing each arc of the tree takes 1 unit of time, then the running time is 11.

b. If both processing elements perform a DFS on their respective halves of tree, the running time is 4 (To find the dark node)

   i. By the definition of Speedup, We have

      1. Speedup = Ts/Tp = 11/4 = 2.75

   ii. and 2.75 is greater than # of processor "2". There is a speedup anomaly. Super linearity effects due to exploratory decomposition. The cause is that the work performed by parallel and serial algorithms is different.

8.

a. AMAT = Hit TimeL1 + Miss RateL1*(Hit TimeL2 + Miss RateL2 * ( Hit TimeL3 + Miss RateL3 * Miss PenaltyL3 ) )
= 5 + 0.1*(10 + 0.25 * (35 + 0.4 * 100))
= 7.875 ns

9.

a. Shared memory computer uses one or more multiple core processors. In this case it is not difficult to construct a shared memory computer, but it is difficult to construct an efficient shared memory computer. For a true shared-memory computer with p processors and a shared memory with b words, where each word to be accessed we need to have (p*b) switches. Which makes the switching network complex and very expensive in this case.

10.

a. The vertices of the d-dimensional hypercube are d-bit binary numbers. Each adjacent vertex will be separated by a single bit difference. If the number of 1 's in a number is even, that number parity is considered as 1 and if the number of 1 's is odd, then the parity is considered as 0. If a vertex has parity 1, then the adjacent vertex will have parity 0 and vice versa.
b. A cycle in a graph is the path when initial and final vertices are the same.
c. If one vertex has parity 1 next vertex will have parity 0, next will have parity 1 , it will continue as such. To reach the final vertex with same parity as initial vertex (both are same here) the cycle length should be even. else the parity will not be same. Therefore, the cycles in a d-dimensional hypercube can't be of odd length.