CSE530 Distributed Systems: Concepts and Design
End-semester Exam (Winter 2024) - May 9, 2024
Max. Time: (2 hours 30 minutes) Max. Marks: 60

VERY VERY IMPORTANT NOTES:

1. Please ensure the question paper has **X printed sides [Y pages].**
2. Calculator is allowed but Sharing of calculators is **NOT ALLOWED**.
3. Laptops and Mobile phones are strictly not allowed.
4. For every question or part of a question, you are required to enter **ONLY** the **FINAL** answer in the space provided immediately after the question.
   a. For numerical type answers, just enter the final number without any units. The units will be mentioned in the question. For example, if your final answer is 30 milliseconds, please enter only **30** in the answer field for that question.
   b. For multiple choice questions, you may have more than one option correct. Please select/write all the correct answers from the options provided for that question. There is no partial marking for partially correct answers.
5. Before starting to answer any question, please read it completely.
6. Exam is fully closed-book. No cheat sheet etc. is allowed.

QUESTION 1 PART 1: In Raft, what happens if a candidate receives an AppendEntries RPC from a leader having a lower term? **[1 mark]**

1. The candidate accepts the AppendEntries RPC and transitions to the follower state immediately.
2. The candidate rejects the AppendEntries RPC and becomes the leader immediately.
3. The candidate rejects the AppendEntries RPC and continues in the candidate state.
4. The candidate rejects the AppendEntries RPC and transitions to the follower state immediately.

Solution: Option 3

sends out heartbeats announcing its leadership.
2. The candidate **C** updates its term and transitions to the follower state.
3. The candidate **C** resends the RequestVote RPC to **Node N** to verify Node N really had a greater term.
4. The candidate **C** simply ignores the rejection from Node N and continues the election.

Solution: Option 2

QUESTION 2 PART 1: What strategies are used by BitTorrent to download chunks? **[1 mark]**

1. Once a sub-piece for a particular piece has been requested, the remaining sub-pieces from that particular piece are requested before sub-pieces from any other piece.
2. When selecting which piece to start downloading next, peers generally download pieces which are present on fewest of the other peers in the network.
3. When selecting which piece to start downloading next, peers generally download pieces which are present on most of the other peers in the network.
4. Pieces to download are selected at random in the beginning until the first complete piece is assembled.

Solution: 1, 2, 4

QUESTION 2 PART 2: Please select the correct statements with regards to BitTorrent? **[1 mark]**

1. Once a peer is done downloading, it stops uploading to other peers since it does not have any requirement of downloading any content.
2. A peer decides which peers to unchoke/choke based on the current download rate offered by the other peers.
3. A peer unchokes a single peer periodically - regardless of the current download rate from it (optimistic unchoke) to explore better download rates.

a "PREPARE-OK" message to the coordinator during the PREPARE phase, but then later fails before the "COMMIT" phase? Note that no information is given in the question about what message is sent by other participants in the PREPARE phase.

1. If any of the other participants have sent a prepare-fail message to the coordinator in the prepare phase, then in the commit phase, the coordinator will send an abort message to all the participants.
2. If majority of the other participants have sent a prepare-ok message to the coordinator in the prepare phase, then in the commit phase, the coordinator will send a commit message to all the participants.
3. If all the other participants have sent a prepare-ok message to the coordinator in the prepare phase, then in the commit phase, the coordinator will send an abort message to all the participants.
4. If all the other participants have sent a prepare-ok message to the coordinator in the prepare phase, then in the commit phase, the coordinator will send a commit message to all the participants.

Solution: Option 1 and Option 4

<u>QUESTION 4 PART 1:</u> Please select the correct statements in the context of Google Borg. [1 mark]

1. It is necessary for all the tasks belonging to the same job to run the same program binary.
2. Different tasks of the same job can run different program binaries.
3. It is necessary for all the tasks belonging to the same job to run in the same Borg cell.
4. Different tasks of the same job can run in different Borg cells.

Solution: Option 1 and Option 3

<u>QUESTION 4 PART 2:</u> Please select the correct statements in the context of Google Borg. [1 mark]

1. Different machines in a Borg cell can have different specifications in terms of CPU, RAM, disk, network etc.
2. All the machines in a Borg cell have same specifications in terms of CPU, RAM, disk, network etc.
3. Majority of the tasks in Google Borg run inside virtual machines.

mark]

1. It is possible to update the resource requirements of a running job in Borg without restarting or resubmitting it.
2. Higher priority tasks can always preempt lower priority tasks.
3. Jobs with insufficient quota are not allowed to enter into the pending job queue.
4. Jobs with sufficient quota are always able to run immediately without spending any time in the pending job queue.

Solution: Any one of the following combinations should be marked:

1. **Option 1 and Option 3 [1 mark]**
2. **Option 1, Option 2 and Option 3 [1 mark]**

QUESTION 4 PART 4: Please select the correct statements in the context of Google Borg. [1 mark]

1. Borgmaster maintains the state of the cell only in memory but not in local disk.
2. Borgmaster maintains the state of the cell only in local disk but not in memory.
3. Borgmaster maintains the state of the cell either in local disk or in memory but not in both.
4. Borgmaster maintains the state of the cell in memory as well as local disk.

Solution: Option 4

QUESTION 4 PART 5: Please select the correct statements in the context of Google Borg. [1 mark]

1. The leader among the Borgmaster replicas is the one which communicates with all the Borglets.
2. Each Borgmaster replica is responsible for communicating with only a portion of the Borglets.
3. If a Borglet does not respond to several poll messages by the BorgMaster replica, that Borglet is marked as down and any tasks it was running are rescheduled on other machines.
4. If the Borglet is alive but not able to contact Borgmaster, it will immediately kill all the currently running tasks as the Borgmaster will anyway reschedule these tasks on some other machine.

Solution: Any one of the following combinations should be marked:

1. **Option 1, Option 2, and Option 3 [1 mark]**
2. **Option 2 and Option 3 [1 mark]**

class.
2. Borg spreads tasks of a job across different machines, racks and power domains.
3. Instead of examining each and every machine for feasibility checking, Borg examines machines in a random order until it has found sufficient number of feasible machines and then selects the best within that set.
4. Borg automatically reschedules the evicted tasks on the same or another machine.

Solution: Option 2, and Option 4

QUESTION 5 PART 1: Please select the correct statements regarding replication Spanner: [1 mark]

1. Spanner allows a tablet to be replicated across multiple data centers in the same geographic location but not across multiple geographic locations.
2. Spanner allows a tablet to be replicated across multiple data centers in the same or different geographic locations.
3. Spanner replicates a tablet across multiple spanservers present in a single Paxos group.
4. Spanner replicates a tablet across multiple spanservers, which may be present in different Paxos groups.

Solution: Option 2 and Option 3

QUESTION 5 PART 2: Please select the correct statements regarding Paxos in Spanner: [1 mark]

1. Spanservers in the same Paxos group can be present in different geographic locations.
2. Spanservers in the same Paxos group can never be present in different geographic locations.
3. Spanner client always selects a Paxos leader as the coordinator leader during two phase commit.
4. Spanner client can select either a Paxos leader or a Paxos follower as the coordinator leader during two-phase commit.

Solution: Option 1 and Option 3

QUESTION 5 PART 3: Please select the correct statements regarding the distributed transactions in Spanner: [1 mark]

3. The spanner client is responsible for selecting the coordinator leader for any distributed transaction.
4. The spanner client has no say in selecting the coordinator leader for any distributed transaction.

Solution: Option 2 and Option 3

<u>QUESTION 5 PART 4:</u> Please select the correct statements regarding reads and writes in Spanner [1 mark]

1. The client necessarily needs to contact a leader for reading any tablet stored in Spanner.
2. The client does not necessarily need to contact a leader for reading any tablet stored in Spanner.
3. The client necessarily needs to contact a leader for writing to any tablet stored in Spanner.
4. The client does not necessarily need to contact a leader for writing to any tablet stored in Spanner.

Solution: Option 2 and Option 3

<u>QUESTION 5 PART 5:</u> Please select the correct statements for read and write requests in Spanner. Note that these requests can be part of any kind of transactions or they can be standalone read/write requests also. [1 mark]

1. The client does not necessarily need to acquire locks for a read request.
2. The client necessarily needs to acquire locks for a read request.
3. The client does not necessarily need to acquire locks for a write request.
4. The client necessarily needs to acquire locks for a write request.

Solution: Option 1 and Option 4

<u>QUESTION 5 PART 6:</u> Please select the correct statements for deciding timestamps in transactions in Spanner. Assume that each transaction necessarily spans across multiple shards. [1 mark]

1. In read-write transactions, the coordinator leader coordinates with all the participating leaders to decide the timestamp for a read-write transaction.
2. In read-write transactions, the coordinator leader itself decides the timestamp of a read-write transaction without any

performed/done among all the participating groups to decide the timestamp for a snapshot transaction.

Solution: Any one of the following combinations should be marked:

1. **Option 1 and Option 4 [1 mark]**
2. **Option 1 and Option 3 [1 mark]**
3. **Option 1, Option 3 and Option 4 [1 mark]**

QUESTION 6 PART 1: Consider a client executing the following python code (TT refers to the TrueTime API):

```
T1 = TT.now()
%% In response, the client receives T1 = [20, 25]
x = TT.after(27)
y = TT.after(33)
z = TT.after(26)
T2 = TT.now()
%% In response, the client receives T2 = [28, 32]
```

Select all the possible combinations of x, y and z that the client can get? **[2 marks]**

a. x = True, y = True, z = True
b. x = True, y = True, z = False
c. x = True, y = False, z = True
d. x = True, y = False, z = False
e. x = False, y = True, z = True
f. x = False, y = True, z = False
g. x = False, y = False, z = True
h. x = False, y = False, z = False

Solution: c, g, h

QUESTION 6 PART 2: Consider a client executing the following python code (TT refers to the TrueTime API):

```
T1 = TT.now()
%% In response, the client receives T1 = [20, 25]
x = TT.before(27)
y = TT.before(33)
z = TT.before(26)
T2 = TT.now()
%% In response, the client receives T2 = [28, 32]
```

Select all the possible combinations of x, y and z that the client can get? **[2 marks]**

a. x = True, y = True, z = True
b. x = True, y = True, z = False
c. x = True, y = False, z = True
d. x = True, y = False, z = False

QUESTION 7: Consider a system with 120 CPUs, 120 GB RAM, and two users, A and B, where user A has 40 pending tasks and user B has 30 pending tasks. Each task of user A has a demand vector <1 CPU, 4 GB> while each task of user B has a demand vector <2 CPUs, 2 GB>. While allocating resources to each user, the system ensures that the following constraints are met:

1. Overall CPU utilisation should not exceed 50% at any point in time.
2. Overall Memory (RAM) utilisation should not exceed 75% at any point in time.
3. No resources are left idle. That is, if there are idle/available resources in the system that can be allocated to any user, then the system will allocate those resources immediately to a particular user as long as the overall CPU and memory utilization does not exceed the aforementioned limits. While doing so, the system will also try to follow the specific scheduling strategy it is supposed to follow.
4. Resource share of a particular user is computed taking into account the specified limit on maximum resource utilization.

QUESTION 7 PART 1: Consider a weighted version of Asset Fairness (AF) scheduling strategy. In this version, the strategy tries to ensure that the sum of resource shares of user A is double the sum of resource shares of user B. Assuming that the system uses this weighted AF version for scheduling, compute the maximum number of tasks for each of the two users (user A and user B) which can simultaneously run in the system. **[3 marks]**

Answer: Any one of the following combinations should be written:

1. Number of tasks for User A (x) = 17, Number of tasks for User B (y) = 11 **[3 marks]**
2. Number of tasks for User A (x) = 17, Number of tasks for User B (y) = 9 **[3 marks]**
3. Number of tasks for User A (x) = 18, Number of tasks for User B (y) = 9 **[2 marks]**
4. Number of tasks for User A (x) = 17, Number of tasks for User B (y) = 10 **[2 marks]**
5. Number of tasks for User A (x) = 16, Number of tasks for User B (y) = 10 **[2 marks]**
6. Number of tasks for User A (x) = 22, Number of tasks for User B (y) = 14 **[1.5**

QUESTION 7 PART 2: Consider a weighted version of Dominant Resource Fairness (DRF) scheduling strategy. In this version, the strategy tries to ensure that the dominant resource share of user A is half the dominant resource share of user B. Assuming that the system uses this weighted DRF version for scheduling, compute the maximum number of tasks for each of the two users (user A and user B) which can simultaneously run in the system. **[3 marks]**

Answer: Any one of the following combinations should be written:

1. Number of tasks for User A (x) = 10, Number of tasks for User B (y) = 25 **[3 marks]**
2. Number of tasks for User A (x) = 9, Number of tasks for User B (y) = 25 **[2 marks]**
3. Number of tasks for User A (x) = 11, Number of tasks for User B (y) = 24 **[2 marks]**
4. Number of tasks for User A (x) = 8, Number of tasks for User B (y) = 26 **[1.5 marks]**
5. Number of tasks for User A (x) = 6, Number of tasks for User B (y) = 27 **[1 marks]**
6. Number of tasks for User A (x) = 6, Number of tasks for User B (y) = 24 **[1 marks]**

QUESTION 7 PART 3: Assuming that the system uses the Competitive Equilibrium from Equal Incomes (CEEI) strategy for scheduling, compute the maximum number of tasks for each of the two users (user A and user B) that can simultaneously run in the system. **[1.5 + 1.5 marks]**

Answer:

1. Number of tasks for User A (x) = 10, Number of tasks for User B (y) = 25 **[3 marks]**

QUESTION 8 Consider a MapReduce job running on a local cluster with 20 mappers (map tasks) and 5 reducers (reduce tasks).

Input data to be processed is of the size 10 GB and is stored in a cloud storage remotely. The upload and download data transfer speed between the cloud storage and each node in the local cluster is 64 MB per second. All nodes in the local cluster can concurrently upload or download from the cloud storage at the aforementioned rate via RPC. All the nodes can read from and write to local disk at the rate of 256 MB per second.

it writes the generated intermediate data in its local disk. The size of intermediate data generated per mapper is same as the size of input data split for that mapper.

Failures in Map Phase: It is possible for a mapper to fail with a probability of 0.4. If a mapper fails, then the master reschedules it instantaneously without any delays. Assume that whenever the mapper fails, it has already read the input data from the remote cloud storage and has completed processing the input data but not started writing the intermediate data into its local disk.

Reducer Phase: Assume that the local cluster has sufficient resources such that all the reducers can run in parallel simultaneously. Reducers start only when all the mappers have finished their execution completely. All the reducers reads their respective intermediate data partitions from the mappers' local disks via RPC. The size of the intermediate data partition per reducer is 2 GB. Once the reducer reads the intermediate data, it takes 4 seconds to process the intermediate data and then write the output data into the cloud storage. The size of output data per reducer is 1 GB.

Failures in Reduce Phase: It is possible for a reducer to fail with a probability of 0.25. If a reducer fails, then the master reschedules it instantaneously without any delays. Assume that whenever the reducer fails, it has already read the intermediate data from the mappers' local disk and has completed processing the intermediate data but not started writing the output data into the remote cloud storage.

Assume that the failures in map and reduce phases occur independently. Use 1 GB = 1024 MB, 1 MB = 1024 KB and so on.

Answer the following (All your answers should be specified upto 1 decimal place):

QUESTION 8 PART 1 Compute the time taken (in seconds) by each mapper for downloading the input data from the remote cloud storage. **[1 mark]**

Answer: 8 seconds

QUESTION 8 PART 2 Compute the time taken (in seconds) by each mapper for writing the intermediate data in its local disk. **[1 mark]**

amount of time taken (in seconds) by each mapper to complete its execution (including the time it takes for the mapper to read the input data and write the intermediate data into the local disk). Consider the failures also. **[2 marks]**

Answer: Any one of the following answers should be written:

1. **20.2 seconds**
2. **15 seconds**
3. **15.8 seconds**
4. **23 seconds**
5. **21 seconds**

QUESTION 8 PART 4 Compute the time taken (in seconds) by each reducer for downloading the intermediate data from the mappers's local disk. **[1 mark]**

Answer: Any one of the following answers should be written:

1. **8 seconds**
2. **32 seconds**
3. **40 seconds**

QUESTION 8 PART 5 Compute the time taken (in seconds) by each reducer for writing the output data back into the remote cloud storage. **[1 mark]**

Answer: Any one of the following answers should be written:

1. **16 seconds**
2. **24 seconds**
3. **28 seconds**
4. **48 seconds**

QUESTION 8 PART 6 Compute the average time taken (in seconds) by each reducer to complete its execution (including the time it takes for the reducer to read the intermediate data from local storage and write the output data back into the cloud storage). Consider the failures also. **[2 marks]**

Answer: Any one of the following answers should be written:

1. **29 seconds**
2. **31 seconds**
3. **32 seconds**
4. **35 seconds**
5. **56 seconds**
6. **61 seconds**
7. **71 seconds**

QUESTION 8 PART 7 Compute the average time taken (in seconds) in completing the entire

4. **56 seconds**
5. **81.2 seconds**
6. **91.2 seconds**

QUESTION 9: Consider a distributed data store (having **3 replicas**) in which each replica maintains the state of the data store as a singular string, supporting two fundamental operations: **appending to the string** and **reading the entire string**. Timestamps, denoted as T_k, reflect the physical time agreed upon by all processes within the system. Each operation is described as follows:

Append Operation:
$A_i$xyz[T_s, T_c] signifies that process P_i appends **xyz** to the system's state, where:

- T_s represents the time at which the append operation was initiated.
- T_c represents the time at which the append operation was completed (i.e. Until T_c, P_i was blocked waiting for the append request to return. At T_c , P_i was notified of the completion of the append request. Note that this does not necessarily mean that the append operation completed on the data store / replica.)

Read Operation:
$R_i$abxyz[T_s, T_c] denotes a read operation by process P_i resulting in the response **abxyz**, where:

- T_s signifies the time at which the read operation was initiated.
- T_c signifies the time at which the read operation was completed, indicating when process P_i received the result **abxyz** of the read operation.

Assume that the data store's state (singular string) starts out empty (NULL). Consider the below operations being performed on the distributed data store with four processes P1, P2, P3, P4. Assume that any operation by any process can contact any replica of the data store.

1. A_1x [1, 5]
2. A_1yz [6, 10]
3. A_2cd [3, 7]
4. A_2a [9, 13]
5. R_3**A** [8, 11]
6. R_3**B** [12, 14]
7. R_4**C** [2, 5.5]
8. R_4**D** [15, 16]

R_4C [2, 5.5] = **x**, R_4D [15, 16] = **xcdyza**

c) R_3A [8, 11] = **xcdyza**, R_3B [12, 14] = **xcdyza**, R_4C [2, 5.5] = **xcd**, R_4D [15, 16] = **xcdyza**
d) R_3A [8, 11] = **cdx**, R_3B [12, 14] = **cdxyz**, R_4C [2, 5.5] = **NULL**, R_4D [15, 16] = **cdxyza**

Solution: b

QUESTION 9 PART 2: Select all possible scenarios for all the read operations (A, B, C, D) which are allowed by linearizability consistency? Please use the correct understanding of linearizability discussed towards the end of the semester. [2 marks]

a) R_3A [8, 11] = **xcdyz**, R_3B [12, 14] = **xcdyza**, R_4C [2, 5.5] = **cd**, R_4D [15, 16] = **xcdyza**
b) R_3A [8, 11] = **xcd**, R_3B [12, 14] = **xcdayz**, R_4C [2, 5.5] = **x**, R_4D [15, 16] = **xcdayz**
c) R_3A [8, 11] = **xcda**, R_3B [12, 14] = **xcdyza**, R_4C [2, 5.5] = **xcd**, R_4D [15, 16] = **xcdayz**
d) R_3A [8, 11] = **cdx**, R_3B [12, 14] = **cdxyz**, R_4C [2, 5.5] = **NULL**, R_4D [15, 16] = **cdxyza**

Solution: b and d

QUESTION 9 PART 3: Select all possible scenarios for all the read operations (A, B, C, D) which are allowed by sequential consistency? [2 marks]

a) R_3A [8, 11] = **xyz**, R_3B [12, 14] = **xyz**, R_4C [2, 5.5] = **NULL**, R_4D [15, 16] = **xyzcd**
b) R_3A [8, 11] = **NULL**, R_3B [12, 14] = **xcdyz**, R_4C [2, 5.5] = **xcd**, R_4D [15, 16] = **xcdyza**
c) R_3A [8, 11] = **NULL**, R_3B [12, 14] = **xyzcd**, R_4C [2, 5.5] = **xcd**, R_4D [15, 16] = **xcdyz**
d) R_3A [8, 11] = **x**, R_3B [12, 14] = **xyzcd**, R_4C [2, 5.5] = **x**, R_4D [15, 16] = **xyzcda**

Solution: Any one of the following combinations should be selected:

1. **a and d [2 marks]**
2. **a, b and d [2 marks]**

QUESTION 9 PART 4: Select all possible scenarios for all the read operations (A, B, C, D) which are allowed by causal consistency? [2 marks]

a) R_3A [8, 11] = **xyzcd**, R_3B [12, 14] = **xyzcda**, R_4C [2, 5.5] = **cdx**, R_4D [15, 16] = **cdxyza**
b) R_3A [8, 11] = **NULL**, R_3B [12, 14] = **xcdyz**, R_4C [2, 5.5] = **cdx**, R_4D [15, 16] = **cdxyza**

QUESTION 9 PART 5: Select all possible scenarios for all the read operations (A, B, C, D) which are allowed by eventual consistency? [2 marks]

a) R_3A [8, 11] = **yzxcda**, R_3B [12, 14] = **cdyza**, R_4C [2, 5.5] = **xcd**, R_4D [15, 16] = **cdaxyz**
b) R_3A [8, 11] = **yzxcda**, R_3B [12, 14] = **cdyza**, R_4C [2, 5.5] = **xcd**, R_4D [15, 16] = **xcdyza**
c) R_3A [8, 11] = **yzxcd**, R_3B [12, 14] = **yzxcda**, R_4C [2, 5.5] = **yzx**, R_4D [15, 16] = **yzxcd**
d) R_3A [8, 11] = **xcd**, R_3B [12, 14] = **cdxyz**, R_4C [2, 5.5] = **cdx**, R_4D [15, 16] = **xcdyza**

Solution: b and d

QUESTION 10: Consider a variant of the Chord Distributed Hash Table (DHT) having the following details:

1. 5-bit identifier to identify a data key or network node (i.e. **m** = 5).
2. Each node/machine in DHT maintains a finger table that stores **m** predecessors instead of **m** successors. Nodes do not have any information about their successors.
3. Data keys are stored in the predecessor of their corresponding identifiers (hash values). That is, a data key with identifier **k** gets assigned to the first node such that this node identifier is equal to or less than **k**. This node is the predecessor of **k**, denoted as **pred(k)**. In other words, pred(k) is the **highest node identifier <= k** among the nodes currently present in the network.
4. The finger table of a node **k** stores **m** predecessors. The **i**th predecessor (**0** <= **i** <= **m-1**) in this table is computed by computing **pred(k - 2^i)**.
5. Nodes with identifiers 4, 11, 19, 27, 30 are the only nodes which are currently part of this Chord DHT network.
6. Just like we used modulo arithmetic in the original Chord DHT to ensure succ(k) is always in the range [0, 31], we use a similar means for ensuring that pred(k) is always in the range [0, 31] for any k in the range [0, 31]. That is, if **pred(k)** (= **k - 2^i) is less than zero, then we subtract it from 32 to get the pred(k) in the range [0, 31].**

QUESTION 10 Part 1: Construct finger tables for each of the five nodes currently present in the Chord network (i.e. for node ids = 4, 11, 19, 27, and 30). For each node, just give us the ordered list of

Node 19 = 11, 11, 11, 11, 30 **[1 mark]**
Node 27 = 19, 19, 19, 19, 11 **[1 mark]**
Node 30 = 27, 27, 19, 19, 11 **[1 mark]**

QUESTION 10 Part 2: Node with id = 19 receives the insertion request for data keys with identifiers 7, 11, 20, 25 and 31. Insert each of these five keys to the above-mentioned Chord DHT. For each insertion, provide the complete sequence of nodes traversed - starting from node 19 till the request comes to the final node where the key should be stored. You are necessarily supposed to use finger tables to minimize the number of nodes traversed for inserting any key. **[1 mark for each data key's sequence]**

Answer:
Key = 7 => 19, 11, 4 **[1 mark]**
Key = 11 => 19, 11 **[1 mark]**
Key = 20 => Any one of the following sequences should be marked:

1. 19, 30, 27, 19 **[1 mark]**
2. 19, 30, 19 **[0.5 mark]**
3. 19 **[0.5 mark]**

Key = 25 => Any one of the following sequences should be marked:

1. 19, 30, 27, 19 **[1 mark]**
2. 19, 30, 19 **[0.5 mark]**
3. 19 **[0.5 mark]**

Key = 31 => Any one of the following sequences should be marked:

1. 19, 11, 4, 30 **[1 mark]**
2. 19, 30 **[0.5 mark]**