Grade 11 Computing

Unit 6 · Clients, Servers and the Web

ANSWER KEY · FOR TEACHERS

Full marks 34

Give the marks for the points in the marking scheme; an answer in other words that says the same is right.

1 Key words

1. server
2. encapsulation
3. port number
4. socket
5. acknowledgment
6. stateless
7. cookie
8. certificate authority

K · 4

Each carries 0.5 mark.

2 Label the diagram

1. SQL
2. Database
3. web server program (HTTPS)
4. operating system

U · 2

Each carries 0.5 mark.

3 Multiple choice

i. **b.** 52 — A · 1

Every router takes 1 off: 64 − 12 = 52. The phone and the server don't change it. If the TTL ever reaches 0, the router that sees it throws the packet away, which stops a packet going round a loop for ever.

ii. **b.** ACK 4801 — A · 1

The acknowledgment names the next byte the receiver wants, and it wants 4,801. It keeps 5,301 to 5,800 until the gap is filled; then it will send ACK 5801.

iii. **b.** Try the same request again later: the server is busy or down for now. — U · 1

5xx codes are the server's problem, and 503 says it is only for now. A 4xx code (400, 404) would mean the request itself needs fixing; 401 would mean signing in.

iv. **c.** `{"district": "Ilam", "tea": true, "rank": null}` — U · 1

JSON needs double quotes, allows no comma after the last item, and writes true, false and null in small letters. True with a capital T is Python, not JSON.

v. **c.** 3NF: driver_name depends on bus_no, which isn't the key — A · 1

Its key is one column, so there can be no partial dependency, and it is in 2NF. But booking_id decides bus_no, and bus_no decides driver_name: a transitive dependency. The driver belongs in a buses table, keyed by bus_no.

4 What does it print?

MARKING SCHEME — U · 3

- ✓ `true` and `null` in small letters, in place of Python's `True` and `None`, with every name and string in double quotes.
- ✓ The tuple written as the array `[1, 14]`, and read back as the list `[1, 14]`.
- ✓ `False`: JSON has no tuples, so the value that came back isn't equal to the one that went out.

Science Playground, Katunje, Bhaktapur
Book a class visit: https://science-playground.com/booking

5 Trace the program

ANSWER **A · 4**

ch	bits	ones	parity	Output
N	1001110	4	0	N 01001110
E	1000101	3	1	E 11000101
P	1010000	2	0	P 01010000

MARKING SCHEME

- ✓ The three codes in binary: N 1001110 (78), E 1000101 (69), P 1010000 (80).
- ✓ `ones` 4, 3 and 2.
- ✓ `parity` 0, 1, 0: the bit that makes the count of 1s even (`ones % 2`).
- ✓ The output: each letter, then its parity bit in front of its 7 bits (`01001110`, `11000101`, `01010000`).

6 Find and fix the bug

MARKING SCHEME **A · 3**

- ✓ Line 5, a KeyError (for `'place'`).
- ✓ 500 from the crash; 400 Bad Request with a reason is what a checked request should get.
- ✓ `value = item.get(key, "")` (or a check that the key is there before reading it).

7 Write a program

ANSWER **A · 4**

```
import json

text = '[{"what": "a pen", "place": "Room 4"}, {"what": "an umbrella", "place":
"the gate"}, {"what": "a ruler", "place": "Room 4"}]'
items = json.loads(text)
counts = {}
for item in items:
    counts[item["place"]] = counts.get(item["place"], 0) + 1
most = ""
for place in counts:
    print(place, counts[place])
    if most == "" or counts[place] > counts[most]:
        most = place
print("Most:", most)
```

```
OUTPUT
Room 4 2
the gate 1
Most: Room 4
```

Other right ways: `max(counts, key=counts.get)` for the place with the most.

MARKING SCHEME

- ✓ The text read with `json.loads()` into a list of dictionaries.
- ✓ A count for each place, kept in a dictionary.
- ✓ Each place printed with its count.
- ✓ The place with the most items found and printed.

Science Playground, Katunje, Bhaktapur
Book a class visit: https://science-playground.com/booking

8 Long answer

ANSWER **A · 4**

a. **Update:** a driver's phone number is written in every ticket for that bus; change it in some rows and not others, and the table gives two numbers. **Insertion:** a new bus, with its route and driver, can't be recorded until someone buys a ticket for it. **Deletion:** deleting the only ticket ever sold for a bus loses which route it runs and who drives it.

b. The table is in 1NF (one value per field, key ticket_no) and, with a single-column key, in 2NF. It isn't in 3NF: route, driver and driver_phone depend on bus_no, and driver_phone on driver, none of which is the key (transitive dependencies). Each group of facts goes to the table of the thing it is about:

- **drivers** (driver_id PK, name, phone)
- **buses** (bus_no PK, route, driver_id FK → drivers)
- **passengers** (passenger_id PK, name, phone)
- **tickets** (ticket_no PK, bus_no FK → buses, travel_date, seat, passenger_id FK → passengers)

Each non-key field now gives a fact about its table's key, the whole key and nothing but the key: 3NF. (The passenger needs some thought: the old table has no passenger ID and names aren't unique, so its passenger_name and passenger_phone could be taken as facts about the ticket. Giving passengers a table of their own, with an ID, writes a regular passenger's phone once. Drivers get an ID for the same reason.)

c. Every foreign key must point to a real row: the database refuses a ticket for a bus_no that isn't in buses, and refuses to delete a bus, a driver or a passenger that tickets still point to. So the data can never contain a ticket on a bus that doesn't exist.

MARKING SCHEME

- ✓ An update anomaly (a driver's phone in many rows, changed in some and not others), an insertion anomaly (a bus can't be recorded until a ticket is sold) and a deletion anomaly (deleting a bus's only ticket loses its route and driver).
- ✓ Not in 3NF: route, driver and driver_phone depend on bus_no, which isn't the key (a transitive dependency); it is in 1NF and 2NF.
- ✓ Tables such as drivers (driver_id PK, name, phone), buses (bus_no PK, route, driver_id FK), passengers (passenger_id PK, name, phone) and tickets (ticket_no PK, bus_no FK, travel_date, seat, passenger_id FK).
- ✓ Referential integrity: every foreign key must point to a real row, so a ticket for a bus that doesn't exist is refused, and so is deleting a bus that tickets still point to.

9 Segments and acknowledgments

MARKING SCHEME **A · 3**

- ✓ The segments as bytes 3001–3800, 3801–4600 and 4601–5400.
- ✓ ACK 3001 after each early segment: the receiver keeps them but still wants byte 3001.
- ✓ 3001–3800 sent again when the timer runs out, then ACK 5401.

10 Think

ANSWER

Because the middle of the network only moves packets, it doesn't need to understand what they carry. A new use needs only programs at the two ends that agree with each other; nobody has to change, or ask permission from, the routers in between. That is the end-to-end principle. The old telephone network kept its cleverness in the exchanges, so every new service needed the network itself to change. The price: the network promises

Science Playground, Katunje, Bhaktapur
Book a class visit: https://science-playground.com/booking

nothing, so packets can be lost, delayed or arrive out of order when links are busy, and the ends must repair that themselves (TCP), or live with it (a video call over UDP).

MARKING SCHEME

- ✓ Routers don't need to understand the data, so a new use needs only new programs at the ends, with no change to the network and no permission (the end-to-end principle).
- ✓ A price: nothing is guaranteed, so packets are lost or delayed when the network is busy and the ends must repair it (or a live call suffers).

11 Activity: Read real requests

(not marked)

WHAT STUDENTS SHOULD FIND

Most requests are GET; a sign-in or a form sends a POST. The page itself comes back as text/html, its pictures as image/..., and the requests the page's own JavaScript sends (fetch or xhr) usually as application/json, often from an address with "api" in it. Many requests answer **304 Not Modified** the second time, because the browser's own copy was still good; a missing file gives 404.

Marks by level	Knowledge	Understanding	Application	Higher ability	**Total**
Marks	4	7	21	2	**34**

Science Playground, Katunje, Bhaktapur
Book a class visit: https://science-playground.com/booking