```
(i). #!/bin/bash

# Get employee name

read -p "Enter employee name: " name

# Get hours worked

read -p "Enter hours worked: " hours

# Get rate per hour

read -p "Enter rate per hour: " rate

(ii). #!/bin/bash

# Calculate basic pay

basic_pay=$(echo "scale=2; $hours *$rate")

(iii). #!/bin/bash

# Function to calculate tax based on basic pay

calculate_tax(){

   local income=$1

   if [ $income -gt 7000]; then

   echo "$(echo "scale =2; $income * 0.25" | bc)"

   elif[ $income -gt 15000]; then

   echo "$(echo "scale=2; $income * 0.15" | bc)"

   else

   echo "0"

   if

}

# Calculate tax
```

```
tax=$(calculate_tax $basic_pay)
```

(iv).
```
#!/bin/bash

# Calculate the net pay

net_pay=$(echo "scale=2; $basic_pay - $tax" | bc)
```

Q2.

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

int main() {

   // File descriptor for the opened file

   int fd;

   // Open the file "message . tx" in read-write mode with create if non-existent flag

   fd = open("message.txt", O_RDWR | O_CREATE, 0664);

   // Check if opening the file was successful

   if (fd == -1){

      perror ("open");

      exit(EXIT_FAILURE);

   }

   // Write "Hello World" to the file

   const char *message = "Hello World\n";

   ssize_t bytes_written = write(fd, message, strlen(message));

   // Check if writing to th file was successful
```

```
if (bytes_written == -1) {
    perror("write");
    close(fd); // Close the file even on error
    exit(EXIT_FAILURE);
}
// Seek to the beginning of the file for reading
lseek(fd, 0, SEEK_SET);
// Allocate a buffer to store the read content
char buffer[100];
// Read content from the file into the buffer
ssize_t bytes_read = read(fd, buffer, sizeof(buffer) -1);
// Check if reading from the file was successful
if (bytes_read == -1) {
    perror("read");
    close(fd); // Close the file even on error
    exit(EXIT_FAILURE);
}
// Add null terminator to the buffer (read doesn't guarantee it)
buffer[bytes_read]= '\0';
// Printf("Read content: %s\n", buffer);
// Close the file
if (close(fd) == -1){
    perror("close");
```

```
        exit(EXIT_FAILURE);
    }
    return EXIT_SUCCESS;
}
```

Q3.

(i).

```
#!/bin/bash
# Prompt the  user for input
read -p “Enter customer ID: “customer_id
read -p “Enter customer Name: “customer_name
read -p “Enter Units Consumed: “units
```

(ii).

```
# Function to calculate base bill based on unit consumption
calculate_base_bill() {
local consumed_units=$1
if [[ $consumed_units -le 199 ]]; then
echo “$(echo “scale=2; $consumed_units  * 120” | bc)”
elif [[ $consumed_units -le 399 ]]; then
echo “$(echo “scale=2; (199 * 120) + (( $consumed_units -199) * 150)” | bc)”
elif [[ $consumed_units -le 599 ]]; then
echo “$(echo “scale=2; (199 * 120) + (200 * 150)+  (($consumed_units -300) * 180)” | bc)”
else
echo “$(echo “scale=2; (199 * 120) + (200 * 150) + (200 * 180) + (( $consumed_units -599)
*200)” | bc)”
if
}
```

(iii).

```
#Calculate base bill
base_bill=$(calculate_base_bill $units)
#Calculate surcharge (applicableonly for bills above400 units)
surcharge=0
if [[ $units -gt 400 ]]; then
surcharge=$(echo “scale=2; $basebill *150” | bc)
if
#Calculate total bill
total_bill=$(echo “scale=2; $base_bill + $surcharge” | bc)
```