CT1OO/G/13764/21

1i)

```
#!/bin/bash
# Prompt user for input
read -p "Enter Employee Name: " employee _name
read -p "Enter Hours Worked: " hours _worked
read -p "Enter Rate per Hour (KSH): " rate _per _hour

# Display confirmation message
echo "Employee details entered:"
echo "  Name: $employee _name"
echo "  Hours Worked: $hours _worked"
echo "  Rate per Hour (KSH): $rate _per _hour"
```

ii)

```
#!/bin/bash
# Prompt user for input
read -p "Enter Employee Name: " employee _name
read -p "Enter Hours Worked: " hours _worked
read -p "Enter Rate per Hour (KSH): " rate _per _hour

# Calculate basic pay
Basic _pay=$((hours _worked * rate _per _hour))

# Display confirmation message
echo "Employee details entered:"
echo "  Name: $employee _name"
echo "  Hours Worked: $hours _worked"
```

```
echo "  Rate per Hour (KSH): $rate_ per _hour"
echo "  Basic Pay (KSH): $basic _pay"
```

iii) #!/bin/bash

```
# Prompt user for input
read -p "Enter Employee Name: " employee _name
read -p "Enter Hours Worked: " hours _worked
read -p "Enter Rate per Hour (KSH): " rate _per _hour

# Function to calculate tax based on basic pay
Calculate _tax() {
  local salary=$1
  local tax _amount=0
   if [[ $salary -le 15000 ]]; then
    tax _amount=0
elif [[ $salary -le 70000 ]]; then
    tax _amount=$((salary * 15 / 100))
  else
    tax _amount=$((salary * 25 / 100))
  fi

  echo $tax _amount
   }

# Calculate tax
Tax _amount=$(calculate _tax $basic _pay)

# Display employee details
```

```
echo "Employee Name: $employee _name"
echo "Basic Pay (KSH): $basic _pay"
echo "Tax (KSH): $tax _amount"
```

iv)

```
#!/bin/bash
# Prompt user for input
read -p "Enter Employee Name: " employee _name
read -p "Enter Hours Worked: " hours _worked
read -p "Enter Rate per Hour (KSH): " rate _per _hour

# Calculate basic pay
Basic _pay=$((hours _worked * rate _per _hour))

# Function to calculate tax based on basic pay
Calculate _tax() {
  local salary=$1
  local tax _amount=0
 if [[ $salary -le 15000 ]]; then
    tax _amount=0
  elif [[ $salary -le 70000 ]]; then
    tax _amount=$((salary * 15 / 100))
  else
    tax _amount=$((salary * 25 / 100))
  fi
echo $tax _amount
}

# Calculate tax
Tax _amount=$(calculate _tax $basic _pay)
```

```
# Calculate net pay
Net _pay=$((basic _pay – tax _amount))

# Display employee details and net pay
echo "Employee Name: $employee _name"
echo "Basic Pay (KSH): $basic _pay"
echo "Tax (KSH): $tax _amount"
echo "Net Pay (KSH): $net _pay"
```

2)

```
#include <stdio.h>   // Standard Input/ Output operations
#include <fcntl.h>   // File control options
#include <unistd.h>  // Symbolic constants and types for POSIX

int main() {
   int file _descriptor;     // File descriptor for the opened file
   const char *filename = "test.txt";  // Filename to be used
   // Open file for writing only, create it if it doesn't exist, truncate it to 0 length if it exists
   File _descriptor = open(filename, O_WRONLY | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR);
   if (file _descriptor < 0) {
      printf("Error opening file");  // Print error message if opening file fails
      return 1;  // Return 1 to indicate error
       }
// Write "Hello World" to the file
   write(file _descriptor, "Hello World\n", 12);  // 12 is the length of "Hello World\n"

   // Close the file
```

```
    close(file _descriptor);

    // Open file for reading only
    File _descriptor = open(filename, O_RDONLY);
    if (file _descriptor < 0) {
        printf("Error opening file");  // Print error message if opening file fails
        return 1;  // Return 1 to indicate error
}
   char buffer[100];  // Buffer to store read content

    // Read content from the file into the buffer
    Ssize _t bytes _read = read(file _descriptor, buffer, sizeof(buffer));
    if (bytes _read < 0) {
printf("Error reading file");  // Print error message if reading file fails
        return 1;  // Return 1 to indicate error
    }
   // Null-terminate the buffer to treat it as a string
    buffer[bytes _read] = '\0';

    // Print the content read from the file
    printf("Content read from file: %s", buffer);
// Close the file
    close(file _descriptor);

    return 0;  // Return 0 to indicate success
}
```

3i)

```
#!/bin/bash
# Prompt user to enter customer ID, customer name, and units consumed
read -p "Enter Customer ID: " customer _id
read -p "Enter Customer Name: " customer _name
read -p "Enter Units Consumed: " units
```

ii)

```
# Function to calculate bill amount based on unit slabs
calculate _bill() {
  local unit _consumed=$1
  local bill _amount=0

  # Up to 199 units
  if [[ $unit _consumed -le 199 ]]; then
    bill _amount=$((unit _consumed * 120))
  else

  # 200 to 399 units
    if [[ $unit _consumed -le 399 ]]; then
      bill _amount=$((199 * 120 + ((unit _consumed - 199) * 150)))
    else

# 400 to 599 units
      if [[ $unit _consumed -le 599 ]]; then
        bill _amount=$((199 * 120 + (200 * 150) + ((unit _consumed - 399) * 180)))
```

```
    else

 # 600 and above units
      Bill _amount=$((199 * 120 + (200 * 150) + (200 * 180) + ((unit _consumed - 599) * 200)))
    fi
  fi
 fi
echo $bill _amount
}
```

iii)

```
#!/bin/bash
# Prompt user for input
read -p "Enter Units Consumed: " units

# Function to calculate bill amount based on unit slabs
Calculate _bill() {
 local unit _consumed=$1
 local bill _amount=0

  # Up to 199 units
 if [[ $unit _consumed -le 199 ]]; then
  bill _amount=$((unit _consumed * 120))
 else

 # 200 to 399 units
if [[ $unit _consumed -le 399 ]]; then
    bill _amount=$((199 * 120 + ((unit _consumed - 199) * 150)))
  else
```

```
 # 400 to 599 units
     If [[ $unit _consumed -le 599 ]]; then
       bill _amount=$((199 * 120 + (200 * 150) + ((unit _consumed - 399) * 180)))
     else

  # 600 and above units
       Bill _amount=$((199 * 120 + (200 * 150) + (200 * 180) + ((unit _consumed - 599) * 200)))
      fi
    fi
  fi
   echo $bill _amount
}

#  Calculate bill amount
Bill _amount=$(calculate _bill $units)

# Display total bill amount
echo  "Total Bill Amount (KSH): $bill  _amount"
```