Python

1) Write a Python program that accepts an integer and determines whether it is greater than 4^4 and which is 4 mod 34. Input: 922 Output: True Input: 914 Output: False Input: 854 Output: True Input: 854 Output: True

```
def check_integer(num):
    greater_than_4_to_the_4th = num > 4 ** 4

    is_4_mod_34 = num % 34 == 4

    return greater_than_4_to_the_4th and is_4_mod_34

num = int(input("Enter an integer: "))

result = check_integer(num)
print(result)
```

2) Write a Python script to concatenate the following dictionaries to create a new one.
Sample Dictionary : dic1={1:10, 2:20} dic2={3:30, 4:40} dic3={5:50,6:60}
Expected Result : {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}

```
dic1 = {1: 10, 2: 20}
dic2 = {3: 30, 4: 40}
dic3 = {5: 50, 6: 60}

result_dict = {k: v for d in (dic1, dic2, dic3) for k, v in d.items()}

print(result_dict)
```

3) Write a Python program to append text to a file and display the text.

```
file_name = "sample.txt"

with open(file_name, 'a') as file:
    text_to_append = input("Enter text to append to the file: ")
    file.write(text_to_append + "\n")

with open(file_name, 'r') as file:
    file_content = file.read()
    print("File Content:")
    print(file_content)
```

4) Write a Python program to find the sequences of one upper case letter followed by lower case letters

```
import re

def find_uppercase_lowercase_sequences(text):
    pattern = r'[A-Z][a-z]+'
    sequences = re.findall(pattern, text)

    if sequences:
        print("Sequences of one uppercase letter followed by lowercase
letters:")
        for seq in sequences:
            print(seq)
    else:
        print("No sequences found in the text.")

text = input("Enter a text: ")

find_uppercase_lowercase_sequences(text)
```

5) Write a Python program that develops a calculator interface with buttons for digits and operators, arranging them in a grid

```
import tkinter as tk

def button_click(event):
    current = entry.get()
    text = event.widget.cget("text")

    if text == "=":
        try:
            result = eval(current)
            entry.delete(0, tk.END)
            entry.insert(0, str(result))
        except Exception as e:
            entry.delete(0, tk.END)
            entry.insert(0, "Error")
    elif text == "C":
        entry.delete(0, tk.END)
    else:
        entry.insert(tk.END, text)

root = tk.Tk()
root.title("Calculator")

entry = tk.Entry(root, font="Helvetica 20")
```

```
entry.grid(row=0, column=0, columnspan=4)

button_labels = [
    "7", "8", "9", "+",
    "4", "5", "6", "-",
    "1", "2", "3", "*",
    "C", "0", "=", "/"
]

row_num = 1
col_num = 0

for label in button_labels:
    button = tk.Button(root, text=label, font="Helvetica 20")
    button.grid(row=row_num, column=col_num, padx=10, pady=10)
    button.bind("<Button-1>", button_click)
    col_num += 1
    if col_num > 3:
        col_num = 0
        row_num += 1

root.mainloop()
```

Data Structures

1) Write a program to store the elements in a 1-D array and perform the operations like searching, sorting and reversing the elements. [Menu Driven]

```c
#include <stdio.h>

void display_menu() {
    printf("\nMenu:\n");
    printf("1. Insert an element\n");
    printf("2. Search an element\n");
    printf("3. Sort elements\n");
    printf("4. Reverse elements\n");
    printf("5. Display elements\n");
    printf("6. Exit\n");
}

void insert_element(int arr[], int *size, int element) {
    arr[*size] = element;
    (*size)++;
    printf("%d added to the array.\n", element);
}

void search_element(int arr[], int size, int element) {
    int found = 0;
    for (int i = 0; i < size; i++) {
        if (arr[i] == element) {
            printf("%d found in the array at index %d.\n", element, i);
            found = 1;
            break;
        }
    }
    if (!found) {
        printf("%d not found in the array.\n", element);
    }
}

void sort_elements(int arr[], int size) {
    for (int i = 0; i < size - 1; i++) {
        for (int j = 0; j < size - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
```

```
    printf("Array sorted in ascending order.\n");
}

void reverse_elements(int arr[], int size) {
    int temp, i, j;
    for (i = 0, j = size - 1; i < j; i++, j--) {
        temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }
    printf("Array reversed.\n");
}

void display_elements(int arr[], int size) {
    printf("Array elements: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main() {
    int arr[100];
    int size = 0;
    int choice, element;

    while (1) {
        display_menu();
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter the element to insert: ");
                scanf("%d", &element);
                insert_element(arr, &size, element);
                break;
            case 2:
                printf("Enter the element to search: ");
                scanf("%d", &element);
                search_element(arr, size, element);
                break;
            case 3:
                sort_elements(arr, size);
                break;
            case 4:
                reverse_elements(arr, size);
                break;
            case 5:
```

```
                display_elements(arr, size);
                break;
            case 6:
                printf("Exiting the program.\n");
                return 0;
            default:
                printf("Invalid choice. Please enter a valid
option.\n");
        }
    }
}
```

2) Read 2 arrays from user and merge them and display the elements in sorted order

```
#include <stdio.h>

void mergeArrays(int arr1[], int size1, int arr2[], int size2, int
result[], int *resultSize) {
    int i, j, k;
    i = j = k = 0;

    while (i < size1 && j < size2) {
        if (arr1[i] < arr2[j]) {
            result[k] = arr1[i];
            i++;
        } else {
            result[k] = arr2[j];
            j++;
        }
        k++;
    }

    while (i < size1) {
        result[k] = arr1[i];
        i++;
        k++;
    }

    while (j < size2) {
        result[k] = arr2[j];
        j++;
        k++;
    }

    *resultSize = k;
}
```

```c
void bubbleSort(int arr[], int size) {
    int temp;
    for (int i = 0; i < size - 1; i++) {
        for (int j = 0; j < size - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

int main() {
    int arr1[100], arr2[100], result[200];
    int size1, size2, resultSize;

    printf("Enter the size of the first array: ");
    scanf("%d", &size1);

    printf("Enter the elements of the first array:\n");
    for (int i = 0; i < size1; i++) {
        scanf("%d", &arr1[i]);
    }

    printf("Enter the size of the second array: ");
    scanf("%d", &size2);

    printf("Enter the elements of the second array:\n");
    for (int i = 0; i < size2; i++) {
        scanf("%d", &arr2[i]);
    }

    mergeArrays(arr1, size1, arr2, size2, result, &resultSize);

    printf("Merged array before sorting:\n");
    for (int i = 0; i < resultSize; i++) {
        printf("%d ", result[i]);
    }

    bubbleSort(result, resultSize);

    printf("\nMerged array after sorting:\n");
    for (int i = 0; i < resultSize; i++) {
        printf("%d ", result[i]);
    }

    return 0;
```

```
}
```

3) Write a program to search the elements in the linked list and display the same

```c
#include <stdio.h>
#include <stdlib.h>

// Define a structure for a linked list node
struct Node {
    int data;
    struct Node* next;
};

// Function to insert a new node at the end of the linked list
void insert(struct Node** head, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;

    if (*head == NULL) {
        *head = newNode;
    } else {
        struct Node* current = *head;
        while (current->next != NULL) {
            current = current->next;
        }
        current->next = newNode;
    }
}

// Function to search for and display elements in the linked list
void searchAndDisplay(struct Node* head, int target) {
    int found = 0;
    struct Node* current = head;

    while (current != NULL) {
        if (current->data == target) {
            printf("%d found in the linked list.\n", target);
            found = 1;
        }
        current = current->next;
    }

    if (!found) {
        printf("%d not found in the linked list.\n", target);
    }
```

```c
}

// Function to display the elements in the linked list
void display(struct Node* head) {
    struct Node* current = head;

    printf("Linked List Elements: ");
    while (current != NULL) {
        printf("%d -> ", current->data);
        current = current->next;
    }
    printf("NULL\n");
}

int main() {
    struct Node* head = NULL;
    int choice, element;

    while (1) {
        printf("\nMenu:\n");
        printf("1. Insert an element\n");
        printf("2. Search and Display an element\n");
        printf("3. Display all elements\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter the element to insert: ");
                scanf("%d", &element);
                insert(&head, element);
                break;
            case 2:
                printf("Enter the element to search: ");
                scanf("%d", &element);
                searchAndDisplay(head, element);
                break;
            case 3:
                display(head);
                break;
            case 4:
                printf("Exiting the program.\n");
                return 0;
            default:
                printf("Invalid choice. Please enter a valid
option.\n");
        }
    }
```

```
    return 0;
}
```

4) Write a program to implement the concept of stack with push pop display and exit operations

```
#include <stdio.h>
#include <stdlib.h>

#define MAX_SIZE 100

struct Stack {
    int items[MAX_SIZE];
    int top;
};

void initializeStack(struct Stack *s) {
    s->top = -1;
}

int isFull(struct Stack *s) {
    return s->top == MAX_SIZE - 1;
}

int isEmpty(struct Stack *s) {
    return s->top == -1;
}

void push(struct Stack *s, int value) {
    if (isFull(s)) {
        printf("Stack is full. Cannot push %d.\n", value);
    } else {
        s->items[++(s->top)] = value;
        printf("Pushed %d onto the stack.\n", value);
    }
}

int pop(struct Stack *s) {
    if (isEmpty(s)) {
        printf("Stack is empty. Cannot pop.\n");
        return -1; // Return a sentinel value
    } else {
        int popped = s->items[(s->top)--];
        return popped;
    }
}
```

```c
void display(struct Stack *s) {
    if (isEmpty(s)) {
        printf("Stack is empty.\n");
    } else {
        printf("Stack elements: ");
        for (int i = 0; i <= s->top; i++) {
            printf("%d ", s->items[i]);
        }
        printf("\n");
    }
}

int main() {
    struct Stack stack;
    initializeStack(&stack);
    int choice, value;

    while (1) {
        printf("\nMenu:\n");
        printf("1. Push\n");
        printf("2. Pop\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter the value to push: ");
                scanf("%d", &value);
                push(&stack, value);
                break;
            case 2:
                if (!isEmpty(&stack)) {
                    int popped = pop(&stack);
                    printf("Popped value: %d\n", popped);
                }
                break;
            case 3:
                display(&stack);
                break;
            case 4:
                printf("Exiting the program.\n");
                return 0;
            default:
                printf("Invalid choice. Please enter a valid
option.\n");
        }
```

```
    }

    return 0;
}
```

5) Write a program to implement the concept of linear probing.

```
#include <stdio.h>
#include <stdlib.h>

#define SIZE 10

struct HashTable {
    int keys[SIZE];
    int values[SIZE];
    int size;
};

void initialize(struct HashTable* table) {
    table->size = 0;
    for (int i = 0; i < SIZE; i++) {
        table->keys[i] = -1;
        table->values[i] = -1;
    }
}

int hash(int key) {
    return key % SIZE;
}

void insert(struct HashTable* table, int key, int value) {
    if (table->size == SIZE) {
        printf("Hash table is full. Cannot insert (%d, %d).\n", key,
value);
        return;
    }

    int index = hash(key);

    while (table->keys[index] != -1) {
        index = (index + 1) % SIZE;
    }

    table->keys[index] = key;
    table->values[index] = value;
    table->size++;
```

```
    printf("Inserted (%d, %d) at index %d.\n", key, value, index);
}

int search(struct HashTable* table, int key) {
    int index = hash(key);

    while (table->keys[index] != -1) {
        if (table->keys[index] == key) {
            return table->values[index];
        }
        index = (index + 1) % SIZE;
    }

    return -1;}

void display(struct HashTable* table) {
    printf("Hash Table Contents:\n");
    for (int i = 0; i < SIZE; i++) {
        if (table->keys[i] != -1) {
            printf("Index %d: Key=%d, Value=%d\n", i, table->keys[i],
table->values[i]);
        }
    }
}

int main() {
    struct HashTable table;
    initialize(&table);

    insert(&table, 2, 20);
    insert(&table, 3, 30);
    insert(&table, 12, 120);
    insert(&table, 4, 40);

    display(&table);

    int key = 12;
    int result = search(&table, key);
    if (result != -1) {
        printf("Value for key %d: %d\n", key, result);
    } else {
        printf("Key %d not found in the hash table.\n", key);
    }

    return 0;
}
```

Operating Systems

1) Write a Linux command:
   a) To show the current working directory

```
pwd
```

   b) To change a directory

```
cd /path/to/directory
```

   c) To create a new directory

```
mkdir directory_name
```

   d) To remove a directory

```
rmdir directory_name
```

   e) To display a content of file

```
cat filename
```

2) Write a Linux command:
   a) To copy a file to some other location

```
cp source_file destination_directory
```

   b) To move a file to different location

```
mv source_file destination_directory
```

   c) To show the difference in content of two files

```
diff file1 file2
```

   d) To count the no. of lines, word, character in the file

```
wc filename
```

   e) To display the unique content of file

```
sort filename | uniq
```

3) Write a shell program to print menu
    a) Display date and time
    b) Display present working directory
    c) Detailed List of files
    d) Who is logged in
    e) To make a new directory

```
#!/bin/bash

while true; do
    clear
    echo "Menu:"
    echo "A) Display date and time"
    echo "B) Display present working directory"
    echo "C) Detailed List of files"
    echo "D) Who is logged in"
    echo "E) Make a new directory"
    echo "Q) Quit"
    read -p "Enter your choice (A/B/C/D/E/Q): " choice

    case $choice in
        [Aa])
            date
            ;;
        [Bb])
            pwd
            ;;
        [Cc])
            ls -l
            ;;
        [Dd])
            who
            ;;
        [Ee])
            read -p "Enter the name of the new directory: " newdir
            mkdir "$newdir"
            echo "Directory '$newdir' created."
            ;;
        [Qq])
            echo "Exiting the menu program."
            exit 0
            ;;
        *)
            echo "Invalid choice. Please select a valid option."
            ;;
    esac

    read -p "Press Enter to continue..."
done
```

4) Write a shell program to take input from the user and print the value on the terminal.

```
#!/bin/bash

read -p "Enter a value: " user_input
echo "You entered: $user_input"
```

5) Write a shell program using basic operators and IF ELSE statement.

```
#!/bin/bash

read -p "Enter the first number: " num1
read -p "Enter the second number: " num2
read -p "Enter an operator (+, -, *, /): " operator

if [ "$operator" == "+" ]; then
    result=$((num1 + num2))
    echo "Result: $num1 + $num2 = $result"
elif [ "$operator" == "-" ]; then
    result=$((num1 - num2))
    echo "Result: $num1 - $num2 = $result"
elif [ "$operator" == "*" ]; then
    result=$((num1 * num2))
    echo "Result: $num1 * $num2 = $result"
elif [ "$operator" == "/" ]; then
    if [ "$num2" -ne 0 ]; then
        result=$(awk "BEGIN {printf \"%.2f\", $num1 / $num2}")
        echo "Result: $num1 / $num2 = $result"
    else
        echo "Error: Division by zero is not allowed."
    fi
else
    echo "Error: Invalid operator. Please use +, -, *, or /."
fi
```

Mobile Programming

1) Write a dart program to Print whether the user is eligible for voting or not.(take name and age as input).

```dart
import 'dart:io';

void main() {
  print("Enter your name: ");
  String name = stdin.readLineSync()!;

  print("Enter your age: ");
  int age = int.parse(stdin.readLineSync()!);

  if (age >= 18) {
    print("$name, you are eligible to vote!");
  } else {
    print("$name, you are not eligible to vote yet.");
  }
}
```

2) Create a calculator in flutter. (Operations – addition, subtraction, multiplication, division.)

```dart
import 'package:flutter/material.dart';

void main() {
  runApp(CalculatorApp());
}

class CalculatorApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: CalculatorScreen(),
    );
  }
}

class CalculatorScreen extends StatefulWidget {
  @override
  _CalculatorScreenState createState() => _CalculatorScreenState();
}

class _CalculatorScreenState extends State<CalculatorScreen> {
  String _output = "0";
  String _input = "";
  double _num1 = 0.0;
  double _num2 = 0.0;
  String _operator = "";
```

```
bool _operatorClicked = false;

void _onButtonPressed(String buttonText) {
  if (buttonText == "C") {
    _clear();
  } else if (buttonText == "+" ||
      buttonText == "-" ||
      buttonText == "x" ||
      buttonText == "/") {
    _onOperatorPressed(buttonText);
  } else if (buttonText == "=") {
    _calculate();
  } else {
    _onDigitPressed(buttonText);
  }
}

void _onOperatorPressed(String operator) {
  if (_operatorClicked) return;

  if (_num1 == 0.0) {
    _num1 = double.parse(_input);
    _operator = operator;
    _operatorClicked = true;
    _output = _input + ' ' + operator;
  } else {
    _output = _output + ' ' + _input + ' ' + operator;
    _calculate();
    _operator = operator;
    _operatorClicked = true;
  }
  _input = '';
}

void _onDigitPressed(String digit) {
  if (_operatorClicked) {
    _input = digit;
    _operatorClicked = false;
  } else {
    _input = _input + digit;
  }
}

void _calculate() {
  _num2 = double.parse(_input);
  double result = 0;

  switch (_operator) {
    case '+':
```

```
        result = _num1 + _num2;
        break;
      case '-':
        result = _num1 - _num2;
        break;
      case 'x':
        result = _num1 * _num2;
        break;
      case '/':
        result = _num1 / _num2;
        break;
    }

    _num1 = result;
    _input = result.toString();
    _output = _input;
  }

  void _clear() {
    _num1 = 0.0;
    _num2 = 0.0;
    _operator = '';
    _input = '';
    _output = '0';
    _operatorClicked = false;
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Simple Calculator'),
      ),
      body: Column(
        children: <Widget>[
          Expanded(
            child: Container(
              alignment: Alignment.bottomRight,
              padding: EdgeInsets.all(24.0),
              child: Text(
                _output,
                style: TextStyle(fontSize: 36.0),
              ),
            ),
          ),
          Column(
            children: <Widget>[
              Row(
                mainAxisAlignment: MainAxisAlignment.center,
```

```
              children: <Widget>[
                _buildButton('7'),
                _buildButton('8'),
                _buildButton('9'),
                _buildButton('/'),
              ],
            ),
            Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: <Widget>[
                _buildButton('4'),
                _buildButton('5'),
                _buildButton('6'),
                _buildButton('x'),
              ],
            ),
            Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: <Widget>[
                _buildButton('1'),
                _buildButton('2'),
                _buildButton('3'),
                _buildButton('-'),
              ],
            ),
            Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: <Widget>[
                _buildButton('C'),
                _buildButton('0'),
                _buildButton('='),
                _buildButton('+'),
              ],
            ),
          ],
        ),
      ],
    ),
  );
}

Widget _buildButton(String buttonText) {
  return Expanded(
    child: InkWell(
      onTap: () {
        setState(() {
          _onButtonPressed(buttonText);
        });
      },
```

```
        child: Container(
          alignment: Alignment.center,
          child: Text(
            buttonText,
            style: TextStyle(fontSize: 24.0),
          ),
        ),
      ),
    );
  }
}
```

3) Create an application in flutter to print the greetings to the user. (take input in TextField)

```
import 'package:flutter/material.dart';

void main() {
  runApp(HelloWorldApp());
}

class HelloWorldApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: HelloWorldScreen(),
    );
  }
}

class HelloWorldScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Hello, World App'),
      ),
      body: Center(
        child: Text(
          'Hello, World!',
          style: TextStyle(fontSize: 24.0),
        ),
      ),
    );
  }
}
```

4) Write a dart program to the Print table (1-10) of the user input number.

```dart
import 'dart:io';

void main() {
  print("Enter a number to generate its multiplication table: ");
  int number = int.parse(stdin.readLineSync()!);

  print("Multiplication table for $number:");
  for (int i = 1; i <= 10; i++) {
    int result = number * i;
    print("$number x $i = $result");
  }
}
```

5) Write a dart program to print menu
   a) Area of square
   b) Perimeter of square
   c) Area of rectangle
   d) Perimeter of rectangle

   And perform the operation as per selected option

```dart
import 'dart:io';

void main() {
  print("Menu:");
  print("1. Area of square");
  print("2. Perimeter of square");
  print("3. Area of rectangle");
  print("4. Perimeter of rectangle");
  print("Enter your choice (1/2/3/4): ");

  int choice = int.parse(stdin.readLineSync()!);

  if (choice == 1) {
    print("Enter the side length of the square: ");
    double side = double.parse(stdin.readLineSync()!);
    double area = side * side;
    print("Area of the square: $area");
  } else if (choice == 2) {
    print("Enter the side length of the square: ");
    double side = double.parse(stdin.readLineSync()!);
    double perimeter = 4 * side;
    print("Perimeter of the square: $perimeter");
  } else if (choice == 3) {
    print("Enter the length of the rectangle: ");
    double length = double.parse(stdin.readLineSync()!);
    print("Enter the width of the rectangle: ");
    double width = double.parse(stdin.readLineSync()!);
    double area = length * width;
```

```
    print("Area of the rectangle: $area");
  } else if (choice == 4) {
    print("Enter the length of the rectangle: ");
    double length = double.parse(stdin.readLineSync()!);
    print("Enter the width of the rectangle: ");
    double width = double.parse(stdin.readLineSync()!);
    double perimeter = 2 * (length + width);
    print("Perimeter of the rectangle: $perimeter");
  } else {
    print("Invalid choice. Please select a valid option (1/2/3/4).");
  }
}
```