B.Sc. (Data Science) II Sem

Data Structures

1. **Write programs to implement the following using an array:**
   a) Stack ADT
   b) Queue ADT

   Stack ADT:

```
class Stack:
    def __init__(self):
        self.stack = []

    def push(self, item):
        self.stack.append(item)

    def pop(self):
        if self.stack:
            return self.stack.pop()
        else:
            print("Stack is empty. Cannot pop.")

    def display(self):
        print("Stack:", self.stack)

if __name__ == "__main__":
    s = Stack()
    while True:
        print("\nOptions:")
        print("1. Push")
        print("2. Pop")
        print("3. Display")
        print("4. Exit")
        choice = input("Enter your choice: ")

        if choice == "1":
            item = input("Enter the item to push: ")
            s.push(item)
        elif choice == "2":
            popped = s.pop()
            if popped is not None:
                print("Popped item:", popped)
        elif choice == "3":
            s.display()
        elif choice == "4":
            print("Exiting...")
            break
        else:
            print("Invalid choice. Please try again.")
```

Queue ADT:

```
class Queue:
    def __init__(self):
        self.queue = []

    def insert(self, item):
        self.queue.append(item)

    def delete(self):
        if self.queue:
            return self.queue.pop(0)
        else:
            print("Queue is empty. Cannot delete.")

    def display(self):
        print("Queue:", self.queue)

if __name__ == "__main__":
    q = Queue()
    while True:
        print("\nOptions:")
        print("1. Insert")
        print("2. Delete")
        print("3. Display")
        print("4. Exit")
        choice = input("Enter your choice: ")

        if choice == "1":
            item = input("Enter the item to insert: ")
            q.insert(item)
        elif choice == "2":
            deleted = q.delete()
            if deleted is not None:
                print("Deleted item:", deleted)
        elif choice == "3":
            q.display()
        elif choice == "4":
            print("Exiting...")
            break
        else:
            print("Invalid choice. Please try again.")
```

2. **Write a program to convert the given infix expression to postfix expression using stack.**

```
def precedence(op):
    if op in ('+', '-'):
        return 1
    elif op in ('*', '/'):
        return 2
    elif op == '^':
        return 3
    return 0

def infix_to_postfix(expr):
    stack = []
    postfix = ""

    for char in expr:
        if char.isalnum():
            postfix += char
        elif char == '(':
            stack.append(char)
        elif char == ')':
            while stack and stack[-1] != '(':
                postfix += stack.pop()
            stack.pop()
        else:
            while stack and precedence(char) <= precedence(stack[-1]):
                postfix += stack.pop()
            stack.append(char)

    while stack:
        postfix += stack.pop()
    return postfix
```

```
# ---- Main program starts here ----
infix = input("Enter an infix expression (e.g., A+B*C): ")
postfix = infix_to_postfix(infix)
print("Postfix expression:", postfix)
```

3. Write a program to evaluate expression using stack.

```
def evaluate_postfix(expression):
    stack = []

    for char in expression:
        if char.isdigit():
            stack.append(int(char))
        else:
            operand2 = stack.pop()
            operand1 = stack.pop()
            if char == '+':
                result = operand1 + operand2
            elif char == '-':
                result = operand1 - operand2
            elif char == '*':
                result = operand1 * operand2
            elif char == '/':
                result = operand1 / operand2
            elif char == '^':
                result = operand1 ** operand2
            else:
                print("Unknown value")
            stack.append(result)

    return stack[0]
```

```
# ---- Main program ----
postfix = input("Enter a postfix expression (e.g., 23*54*+9-): ")
res = evaluate_postfix(postfix)
print("Result=",res)
```

4. Write a program to ensure the parenthesis are nested correctly in an Arithmetic Expression

```
def is_parenthesis_balanced(expression):
    stack = []
    for char in expression:
        if char == '(':
            stack.append(char)
        elif char == ')':
            if stack[-1] != '(':
                return False
            stack.pop()
    return len(stack) == 0
expr = input("Enter an expression ")
if is_parenthesis_balanced(expr):
    print("The parentheses are correctly nested.")
else:
    print("The parentheses are not correctly nested.")
```

5. Write a program to find following using Recursion

(a) Factorial of a number

(b) N^{th} term of the Fibonacci Sequence

(c) GCD of two positive integers

Factorial:

```
def fact(n):
    if n < 0:
        return
    elif n == 0 or n == 1:
```

```
        return 1
    else:
        return n * fact(n - 1)

n = int(input("Enter a number: "))
factorial = fact(n)
print(f"The factorial of {n} is: {factorial}")
```

Fibonacci Sequence:

```
def fibonacci(n):
    if n <= 1:
        return n
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)
n = int(input("How many terms? "))
if n <= 0:
    print("Please enter a positive integer")
else:
    print("Fibonacci Sequence:")
    for i in range(n):
        print(fibonacci(i))
```

GCD:

```
def gcd(a, b):
    if b == 0:
        return a
    else:
        return gcd(b, a % b)

num1 = int(input("Enter the first positive integer: "))
num2 = int(input("Enter the second positive integer: "))
```

```
if num1 <= 0 or num2 <= 0:
    print("Please enter only positive integers.")
else:
    result = gcd(num1, num2)
    print(f"The GCD of {num1} and {num2} is {result}")
```

10. Write a program for sorting the given list numbers in ascending order using the following technique: Bubble sort and Selection sort

Bubble Sort:

```
def bubble_sort(arr):
    n = len(arr)
    for j in range(n):
        for k in range(0, n - j - 1):
            if arr[k] > arr[k + 1]:
                arr[k], arr[k + 1] = arr[k + 1], arr[k]
    return arr
n = int(input("Enter the size of N: "))
a = []
print(f"Enter {n} elements:")
for i in range(n):
    element = int(input())
    a.append(element)
sorted_array = bubble_sort(a)
print("Bubble Sorted Elements:")
for num in sorted_array:
    print(num)
```

Selection Sort:

```
def selection_sort(arr):
    n = len(arr)
    for i in range(n):
        min_index = i
        for j in range(i + 1, n):
```

```
            if arr[j] < arr[min_index]:
                min_index = j
        arr[i], arr[min_index] = arr[min_index], arr[i]
    return arr
n = int(input("Enter the size of N: "))
a = []
print(f"Enter {n} elements:")
for i in range(n):
    element = int(input())
    a.append(element)
sorted_array = selection_sort(a)
print("Selection Sorted Elements:")
for num in sorted_array:
    print(num)
```