Qno.1)

def srt(stk):

temp\_stk = []

while stk:

top\_element = stk.pop()

while temp\_stk and temp\_stk[-1] < top\_element:

stk.append(temp\_stk.pop())

temp\_stk.append(top\_element)

# Copy the sorted elements back to the original stack

while temp\_stk:

stk.append(temp\_stk.pop())

return stk

stk = ['d', 'a', 'c', 'b']

sorted\_stk = srt(stk)

print(sorted\_stk)

Qno.2)

def next\_greater\_elements(stk):

result = [-1] \* len(stk)

temp\_stk = []

for i, value in enumerate(stk):

while temp\_stk and stk[temp\_stk[-1]] < value:

index = temp\_stk.pop()

result[index] = value

temp\_stk.append(i)

return result

stk = [5, 3, 2, 10, 6, 8, 1, 4, 12, 7, 4]

next\_greater = next\_greater\_elements(stk)

for i in range(len(stk)):

print(f"{stk[i]} -> {next\_greater[i] if next\_greater[i] != -1 else 'none'}")

Qno.3)

def find\_consecutive\_pairs(stk):

consecutive\_pairs = []

for i in range(len(stk) - 1):

if stk[i] + 1 == stk[i + 1]:

consecutive\_pairs.append((stk[i], stk[i + 1]))

return consecutive\_pairs

stk1 = [6, 5, 10, 11, -3, -2, 5, 4]

stk2 = [3, 4, 7, 6, 6, 4]

pairs1 = find\_consecutive\_pairs(stk1)

pairs2 = find\_consecutive\_pairs(stk2)

print("Stack 1:")

for pair in pairs1:

print(pair)

print("Stack 2:")

for pair in pairs2:

print(pair)

Qno.4)

def rewrite\_expression(expr):

stk = []

sign = 1

result = ""

for char in expr:

if char == '(':

stk.append(sign)

elif char == ')':

stk.pop()

elif char == '+':

result += ' ' + ('+' if not stk or stk[-1] == sign else '-') + ' '

elif char == '-':

result += ' ' + ('-' if not stk or stk[-1] == sign else '+') + ' '

else:

result += char

return result

expr1 = "x - (y + z)"

expr2 = "x - (y - z - (u+v)) - w"

new\_expr1 = rewrite\_expression(expr1)

new\_expr2 = rewrite\_expression(expr2)

print(new\_expr1) # Output: "x - y - z"

print(new\_expr2) # Output: "x - y + z + u + v - w"