Lab04

February 15, 2023

1 Questions

Lab04

1. Write a program to circulate values of N variable.
2. Write a program to find square root of a number using Newtons method.
3. Write a program to find exponentiation(power) of a number.
4. Write a program to find linear search in a list.
5. Write a program to find binary search in a list.
6. Write a program to implement selection sort.
7. Write a program to implement merge sort.
8. Write a program to find first N prime number,
9. Write a program to multiply two matrices
10. Write a program to find maximum of a list of a number.
11. Write a program that encrypts a message by adding a key value to every character (Caesar Cipher)
12. Write a program to find whether a given character is present in a string or not.
13. Write a program that uses lambda function to multiply two numbers.
14. Write a program to reverse a string without using recursion.
15. Write a program that counts the amount of lowercase. Uppercase and digit entered by user.

2 Answers

Name: Dipankar Das

Roll 20051554

2.1 Write a program to circulate values of N variable

[1]:
```
n = int(input("Enter the value for N"))
arr = [0 for _ in range (n)]
for i in range(0,n):
  arr[i] = int(input("Enter the number"))
print(arr)
```

```
[23, 34, 1234, 345]
```

2.2 Write a program to find square root of a number using Newtons method

[3]:
```
def squareRoot(n, l):

  x = n

  count = 0

  while (1):
    count += 1
    root = 0.5 * (x + (n / x))
    if (abs(root - x) < l):
        break
    x = root
  return root
# 0.00001
print(squareRoot(int(input("Enter the number")), float(input("Enter the␣
 ↪precessions"))))
```

```
18.083141320042877
```

2.3 Write a program to find exponentiation(power) of a number

[6]:
```
def findPower(base,power)->int:
  if power == 0:
    return 1
  if (power&1) == 1:
    return base*findPower(base, power-1)
  else:
    return base*base*findPower(base, power-2)

print(findPower(int(input("Enter base")), int(input("Enter power"))))
```

```
32
```

2.4 Write a program to find linear search in a list.

[2]:
```
list = [3,424,5345,12,23424]
found = False
search = int(input("Enter the ele to search"))
for i in list:
  if i == search:
    found = True
    print("Found!")
if not found:
  print("Not found")
```

```
Not found
```

2.5 Write a program to find binary search in a list.

[3]:
```
def binary_search(arr, target):
  left, right = 0, len(arr) - 1
  while left <= right:
    mid = (left + right) // 2
    if arr[mid] == target:
      return mid
    elif arr[mid] < target:
      left = mid + 1
    else:
      right = mid - 1
  return -1  # Target not found in the list


arr = [1, 3, 5, 7, 9]
target = 7
index = binary_search(arr, target)
if index != -1:
  print(f"Target {target} found at index {index}")
else:
  print(f"Target {target} not found in the list")
```

```
Target 7 found at index 3
```

2.6 Write a program to implement selection sort

[4]:
```
def selection_sort(arr):
  n = len(arr)
  for i in range(n-1):
    min_idx = i
    for j in range(i+1, n):
      if arr[j] < arr[min_idx]:
        min_idx = j
    arr[i], arr[min_idx] = arr[min_idx], arr[i]
  return arr

my_list = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
sorted_list = selection_sort(my_list)
print(sorted_list)
```

```
[1, 1, 2, 3, 3, 4, 5, 5, 5, 6, 9]
```

2.7 Write a program to implement merge sort

[ ]:
```python
def merge_sort(arr):
  if len(arr) <= 1:
    return arr
  mid = len(arr) // 2
  left = arr[:mid]
  right = arr[mid:]
  left = merge_sort(left)
  right = merge_sort(right)
  return merge(left, right)

def merge(left, right):
  result = []
  i, j = 0, 0
  while i < len(left) and j < len(right):
    if left[i] <= right[j]:
      result.append(left[i])
      i += 1
    else:
      result.append(right[j])
      j += 1
  result += left[i:]
  result += right[j:]
  return result

my_list = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
sorted_list = merge_sort(my_list)
print(sorted_list)
```

2.8 Write a program to find first N prime number

[6]:
```python
def is_prime(n):
  if n <= 1:
      return False
  for i in range(2, int(n**0.5) + 1):
      if n % i == 0:
        return False
  return True

def first_n_primes(n):
  primes = []
  i = 2
  while len(primes) < n:
    if is_prime(i):
      primes.append(i)
    i += 1
```

```
  return primes

print(first_n_primes(10))
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
```

2.9 Write a program to multiply two matrices

[7]:
```
def multiply_matrices(a, b):
    """
    Multiplies two matrices and returns the result.
    """
    # Get the dimensions of the matrices
    a_rows, a_cols = len(a), len(a[0])
    b_rows, b_cols = len(b), len(b[0])

    # Check if the matrices are compatible for multiplication
    if a_cols != b_rows:
        raise ValueError("Matrices are not compatible for multiplication")

    # Create an empty result matrix
    result = [[0 for _ in range(b_cols)] for _ in range(a_rows)]

    # Multiply the matrices
    for i in range(a_rows):
        for j in range(b_cols):
            for k in range(a_cols):
                result[i][j] += a[i][k] * b[k][j]

    return result

a = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
b = [[9, 8, 7], [6, 5, 4], [3, 2, 1]]
c = multiply_matrices(a, b)
print(c)
```

```
[[30, 24, 18], [84, 69, 54], [138, 114, 90]]
```

2.10 Write a program to find maximum of a list of a number.

[9]:
```
numbers = [5, 10, 15, 20, 25]

maximum = max(numbers)

print("The maximum number is:", maximum)
```

```
The maximum number is: 25
```

2.11 Write a program that encrypts a message by adding a key value to every character (Caesar Cipher)

[1]:
```
message = input("Enter your message: ")
key = int(input("Enter the key value: "))

encrypted_message = ""
message = message.upper()

for character in message:
    # Check if the character is a letter
    if character.isalpha():
        # Convert the character to its ASCII code, add the key value, and␣
 ↪convert back to a character
        encrypted_character = chr((ord(character) + key - 65) % 26 + 65)
    else:
        # Non-letter characters are not encrypted
        encrypted_character = character

    encrypted_message += encrypted_character

print("Encrypted message:", encrypted_message)
```

```
Encrypted message: C
```

2.12 Write a program to find whether a given character is present in a string or not.

[6]:
```
str = input("Enter the string")
char = input("Enter the character")
for i in str:
  if i == char:
    print("Found "+i)
```

```
Found b
Found b
```

2.13 Write a program that uses lambda function to multiply two numbers

[7]:
```
multiply = lambda x, y: x * y
result = multiply(3, 5)
print(result)
```

```
15
```

2.14 Write a program to reverse a string without using recursion.

[10]:
```
original_str = "Hello, World!"
reversed_str = original_str[::-1]

print(reversed_str)  # Output: "!dlroW ,olleH"
```

```
!dlroW ,olleH
```

2.15 Write a program that counts the amount of lowercase. Uppercase and digit entered by user

[11]:
```
input_str = input("Enter a string: ")

lowercase_count = 0
uppercase_count = 0
digit_count = 0

for char in input_str:
    if char.islower():
        lowercase_count += 1
    elif char.isupper():
        uppercase_count += 1
    elif char.isdigit():
        digit_count += 1

print("Lowercase count:", lowercase_count)
print("Uppercase count:", uppercase_count)
print("Digit count:", digit_count)
```

```
Lowercase count: 2
Uppercase count: 2
Digit count: 0
```