|  |
| --- |
| **Netaji Subhash Engineering College**  **Department of Computer Science & Engineering**  **B. Tech CSE 2nd Year 3rd Semester**  **2023-2024**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Name of the Course: IT Workshop (Python)**  **Course Code: PCC-CS393**  **Name of the Student: ARITTRA BAG**  **Class Roll No.: 103**  **University Roll No.: 10900122105**  **Date of Experiment: 19/10/2023**  **Date of Submission: 03/11/2023**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Assignment No.: PROJECT\_01**  **Problem Statement:**  **Consider the following series: 1,1,2,3,4,9,8,27,16,81,32,243,64,729,128,2187…This series is a mixture of 2 series. Write a program to find the nth term in the series. The nth term calculated by the program should be printed on the screen. No other character/string or message should be printed besides the value of the nth term.**  **Python Code:**  def find\_nth\_term(n):      if n==0:          return("Invalid Term!")      elif n % 2 == 0:          return (f"The term is: {3 \*\* (n // 2 - 1)}")      else:          return (f"The term is: {2 \*\* (n // 2)}")  print(find\_nth\_term(int(input("Enter the no. of terms: "))))  **Sample Output(s):**  Enter the no. of terms: 5  The term is: 4.  **Assignment No.: PROJECT\_02**  **Problem Statement:**  **Write a Python program that creates a menu-driven sorting algorithm application.**  **Python Code:**  def bubble\_sort(arr):      n = len(arr)      for i in range(n - 1):          for j in range(0, n - i - 1):              if arr[j] > arr[j + 1]:                  arr[j], arr[j + 1] = arr[j + 1], arr[j]  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]  def insertion\_sort(arr):      n = len(arr)      for i in range(1, n):          key = arr[i]          j = i - 1          while j >= 0 and key < arr[j]:              arr[j + 1] = arr[j]              j -= 1          arr[j + 1] = key  def merge\_sort(arr):      if len(arr) > 1:          mid = len(arr) // 2          left\_half = arr[:mid]          right\_half = arr[mid:]          merge\_sort(left\_half)          merge\_sort(right\_half)          i = j = k = 0          while i < len(left\_half) and j < len(right\_half):              if left\_half[i] < right\_half[j]:                  arr[k] = left\_half[i]                  i += 1              else:                  arr[k] = right\_half[j]                  j += 1              k += 1          while i < len(left\_half):              arr[k] = left\_half[i]              i += 1              k += 1          while j < len(right\_half):              arr[k] = right\_half[j]              j += 1              k += 1  def quick\_sort(arr):      if len(arr) <= 1:          return arr      else:          pivot = arr[0]          less\_than\_pivot = [x for x in arr[1:] if x <= pivot]          greater\_than\_pivot = [x for x in arr[1:] if x > pivot]          return quick\_sort(less\_than\_pivot) + [pivot] + quick\_sort(greater\_than\_pivot)  while True:      print("\nChoose a sorting algorithm:\n1. Bubble Sort\n2. Selection Sort\n3. Insertion Sort\n4. Merge Sort\n5. Quick Sort\n6. Exit")      choice = int(input("Enter your choice: "))      if choice == 6:          print("Exiting...")          break      elif choice not in range(1, 6):          print("Invalid choice!")      else:          input\_list = list(map(int, input("Enter a list of numbers separated by spaces: ").split()))          if choice == 1:              bubble\_sort(input\_list)          elif choice == 2:              selection\_sort(input\_list)          elif choice == 3:              insertion\_sort(input\_list)          elif choice == 4:              merge\_sort(input\_list)          elif choice == 5:              input\_list = quick\_sort(input\_list)          print("Sorted list:", input\_list)  **Sample Output(s):**  Choose a sorting algorithm:  1. Bubble Sort  2. Selection Sort  3. Insertion Sort  4. Merge Sort  5. Quick Sort  6. Exit  Enter your choice: 3  Enter a list of numbers separated by spaces: 10 50 30 45 89 -35  Sorted list: [-35, 10, 30, 45, 50, 89]  **Assignment No.: PROJECT\_03**  **Problem Statement:**  **Write a Python program that creates a menu-driven number base converter.**  **Python Code:**  def convert(base\_from, base\_to, num):      try:          if base\_from == 10 and base\_to == 2:              result = bin(int(num))[2:]          elif base\_from == 2 and base\_to == 10:              result = str(int(num, 2))          elif base\_from == 10 and base\_to == 8:              result = oct(int(num))[2:]          elif base\_from == 8 and base\_to == 10:              result = str(int(num, 8))          elif base\_from == 10 and base\_to == 16:              result = hex(int(num))[2:]          elif base\_from == 16 and base\_to == 10:              result = str(int(num, 16))          else:              result = "Invalid conversion"          return result      except ValueError:          return "Invalid input"  while True:      base\_choices = {          1: (10, 2),          2: (2, 10),          3: (10, 8),          4: (8, 10),          5: (10, 16),          6: (16, 10)      }      print("\nNumber Base Converter\n1. Decimal to Binary\n2. Binary to Decimal\n3. Decimal to Octal\n4. Octal to Decimal\n5. Decimal to Hexadecimal\n6. Hexadecimal to Decimal\n7. Exit")      choice = int(input("Enter your choice: "))      if choice == 7:          print("Exiting...")          break      if choice not in range(1, 7):          print("Invalid choice!")      num = input(f"Enter the number in base {base\_choices[choice][0]}: ")      base\_from, base\_to = base\_choices[choice]      result = convert(base\_from, base\_to, num)      print(f"Converted number: {result}")  **Sample Output(s):**    **Number Base Converter**  **1. Decimal to Binary**  **2. Binary to Decimal**  **3. Decimal to Octal**  **4. Octal to Decimal**  **5. Decimal to Hexadecimal**  **6. Hexadecimal to Decimal**  **7. Exit**  **Enter your choice: 2**  **Enter the number in base 2: 1010**  **Converted number: 10**  **Number Base Converter**  **1. Decimal to Binary**  **2. Binary to Decimal**  **3. Decimal to Octal**  **4. Octal to Decimal**  **5. Decimal to Hexadecimal**  **6. Hexadecimal to Decimal**  **7. Exit**  **Enter your choice: 5**  **Enter the number in base 10: 15**  **Converted number: f**  **Assignment No.: PROJECT\_04**  **Problem Statement:**  **Write a Python program to implement stack and queue using a linked list.**  **Python Code:**  class Node:      def \_\_init\_\_(self, data):          self.data = data          self.next = None  class Stack:      def \_\_init\_\_(self):          self.top = None      def push(self, data):          new\_node = Node(data)          new\_node.next = self.top          self.top = new\_node      def pop(self):          if self.top is None:              return None          data = self.top.data          self.top = self.top.next          return data      def display(self):          if self.top is None:              print("Stack is empty!")          else:              print("Stack:")              current = self.top              stack\_items = []              while current:                  stack\_items.append(current.data)                  current = current.next              for item in stack\_items:                  print(item)  class Queue:      def \_\_init\_\_(self):          self.front = self.rear = None      def enqueue(self, data):          new\_node = Node(data)          if self.rear is None:              self.front = self.rear = new\_node              return          self.rear.next = new\_node          self.rear = new\_node      def dequeue(self):          if self.front is None:              return None          data = self.front.data          self.front = self.front.next          if self.front is None:              self.rear = None          return data      def display(self):          if self.front is None:              print("Queue is empty!")          else:              print("Queue:",end="")              current = self.front              queue\_items = []              while current:                  queue\_items.append(current.data)                  current = current.next              print(" ".join(queue\_items))  def main():      stack = Stack()      queue = Queue()      while True:          print("\nMenu:\n1. Push (Stack)\n2. Pop (Stack)\n3. Enqueue (Queue)\n4. Dequeue (Queue)\n5. Display (Stack)\n6. Display (Queue)\n7. Exit")          choice = int(input("Enter your choice: "))          if choice == 1:              data = input("Enter data to push: ")              stack.push(data)          elif choice == 2:              data = stack.pop()              if data is not None:                  print("Popped data:", data)              else:                  print("Stack Underflow!")          elif choice == 3:              data = input("Enter data to enqueue: ")              queue.enqueue(data)          elif choice == 4:              data = queue.dequeue()              if data is not None:                  print("Dequeued data:", data)              else:                  print("Queue Underflow!")          elif choice == 5:              stack.display()          elif choice == 6:              queue.display()          elif choice == 7:              print("Exiting...")              break          else:              print("Invalid choice. Please try again.")    if \_\_name\_\_ == "\_\_main\_\_":      main()  **Sample Output(s):**  Menu:  1. Push (Stack)  2. Pop (Stack)  3. Enqueue (Queue)  4. Dequeue (Queue)  5. Display (Stack)  6. Display (Queue)  7. Exit  Enter your choice: 1  Enter data to push: 10  Menu:  1. Push (Stack)  2. Pop (Stack)  3. Enqueue (Queue)  4. Dequeue (Queue)  5. Display (Stack)  6. Display (Queue)  7. Exit  Enter your choice: 1  Enter data to push: 20  Menu:  1. Push (Stack)  2. Pop (Stack)  3. Enqueue (Queue)  4. Dequeue (Queue)  5. Display (Stack)  6. Display (Queue)  7. Exit  Enter your choice: 1  Enter data to push: 30  Menu:  1. Push (Stack)  2. Pop (Stack)  3. Enqueue (Queue)  4. Dequeue (Queue)  5. Display (Stack)  6. Display (Queue)  7. Exit  Enter your choice: 5  Stack:  30  20  10  Menu:  1. Push (Stack)  2. Pop (Stack)  3. Enqueue (Queue)  4. Dequeue (Queue)  5. Display (Stack)  6. Display (Queue)  7. Exit  Enter your choice: 2  Popped data: 30  Menu:  1. Push (Stack)  2. Pop (Stack)  3. Enqueue (Queue)  4. Dequeue (Queue)  5. Display (Stack)  6. Display (Queue)  7. Exit  Enter your choice: 6  Queue is empty!  **Assignment No.: PROJECT\_05**  **Problem Statement:**  **Write a Python program to build a secure password generator.**  **Python Code:**  import random  import string  def generate\_password(length):      characters = string.ascii\_letters + string.digits + "@#$%&"      password = ''.join(random.choice(characters) for \_ in range(length))      return password  def is\_valid(password):      return any(c.islower() for c in password) and any(c.isupper() for c in password) and any(c.isdigit() for c in password) and any(c in "@#$%&" for c in password)  if \_\_name\_\_ == "\_\_main\_\_":      while True:          length = int(input("Enter the desired password length: "))          while True:              password = generate\_password(length)              if is\_valid(password):                  break          print("Generated Password:", password)          another = input("Generate another password? (y/n): ").lower()          if another != "y":              break  **Sample Output(s):**  Enter the desired password length: 8  Generated Password: 0W@4rCV@  Generate another password? (y/n): Y  Enter the desired password length: 5  Generated Password: B$pj5 |