**Laxmi Charitable Trust’s**

**Sheth L.U.J College of Arts & Sir M.V. College of Science and Commerce**

Department of Information Technology (Bsc.IT Semester IV)

Data Structures Practical

Practical - I

|  |  |
| --- | --- |
| Roll No.: S022 | Name: Harshit Jaiswal |
| Class: SYIT | Batch: B1 |
| Date of Assignment: 01-07-2024 | Date/Time of Submission: |

**Implement the following:**

**Q1. Write a program to store the elements in 1-D array and perform the operations like searching, sorting and reversing the elements. [Menu Driven]**

***Algorithm:***

**Step 1: Initialize an empty array array.**

**Step 2: Print the menu:**

**2.a: Insertion Element (1)**

**2.b: Search Element (2)**

**2.c: Sort Element (3)**

**2.d: Reverse Element (4)**

**2.e: Display Element (5)**

**2.f: Exit Element (6)**

**Step 3: Define six functions:**

**3.a: insert\_element(array):**

**- Prompt user to enter the number of elements to insert.**

**- For each element, prompt user to enter the number using “format” and append it to the array.**

**- Print the updated array.**

**3.b: search\_element(array):**

**- Prompt user to enter the element to search for.**

**- Check if the element is in the array.**

**- If found, print the index (1-indexed).**

**- If not found, print "Element not found".**

**3.c: sort\_element(array):**

**- Sort the array in ascending order.**

**- Print the sorted array.**

**3.d: reverse\_element(array):**

**- Reverse the array.**

**- Print the reversed array.**

**3.e: display\_element(array):**

**- Print the array.**

**3.f: exit\_element(array):**

**- Print "Menu exited".**

**Step 4: Enter a while loop that continues until the user chooses to exit.**

**Step 5: Inside the loop:**

**- Prompt user to enter their choice (1-6).**

**- Based on the user's input, call the appropriate function to perform the desired operation.**

**- If the user enters an invalid input, print "Invalid Input."**

**Step 6: Print "Harshit Jaiswal 22" for student identity.**

***Code:***

**array = []**

**print("\nMenu")**

**print("1. Insertion Element")**

**print("2. Search Element")**

**print("3. Sort Element")**

**print("4. Reverse Element")**

**print("5. Display Element")**

**print("6. Exit Element")**

**def insert\_element(array):**

**print("Performing Insertion Element.")**

**num1 = int(input("Enter the number of elements: "))**

**for i in range (0,num1):**

**num2 = int(input("Enter element {}: ".format(i+1)))**

**array.append(num2)**

**print(array)**

**def search\_element(array):**

**print("Performing Search Element.")**

**element = int(input("Enter your element to find : "))**

**if element in array:**

**print("Element found in the array at index", (array.index(element))+1)**

**else:**

**print("Element not found")**

**def sort\_element(array):**

**print("Performing Sort Element.")**

**array.sort()**

**print("Sorted Element : ",array)**

**def reverse\_element(array):**

**print("Performing Reverse Element.")**

**array.reverse()**

**print("Reversed Element : ",array)**

**def display\_element(array):**

**print("Performing Display Element.")**

**print("The array is : ", array)**

**def exit\_element(array):**

**print("Performing Exit Element.")**

**print("Menu exited")**

**array = []**

**while True:**

**ch = int(input("Enter your choice : "))**

**if ch == 1:**

**insert\_element(array)**

**elif ch == 2:**

**search\_element(array)**

**elif ch == 3:**

**sort\_element(array)**

**elif ch == 4:**

**reverse\_element(array)**

**elif ch == 5:**

**display\_element(array)**

**elif ch == 6:**

**exit\_element(array)**

**break**

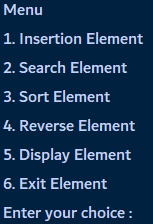
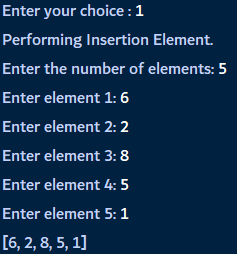
**else:**

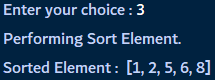
**print("Invalid Input.")**

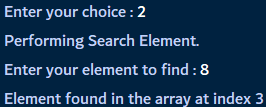
**print("Harshit Jaiswal 22")**

***Output:***

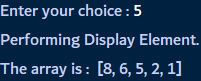
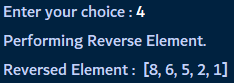
1. **Menu 2. Insertion Element**

****

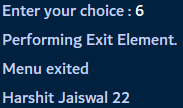
1. **Search Element 4. Sort Element**

****

1. **Sort Element**
2. **Reverse Element 6. Display Element**

****

1. **Exit Element**

****

**Q.2. Read the two arrays from the user and merge them and display the elements in sorted order. [Menu Driven]**

***Algorithm:***

**Step 1: Display Menu**

**- Print the menu options to the user:**

**1. Insertion Element**

**2. Merge Element**

**3. Sort Element**

**4. Display Element**

**5. Exit**

**Step 2: Get User Choice**

**- Ask the user to enter their choice (1-5)**

**Step 3: Perform Action Based on User Choice**

**- If user choice is 1:**

**- Call insert\_element() function**

**- If user choice is 2:**

**- Call merge\_element() function**

**- If user choice is 3:**

**- Call sort\_element() function**

**- If user choice is 4:**

**- Call display\_element() function**

**- If user choice is 5:**

**- Print "Exiting Menu" and exit the program**

**- If user choice is invalid:**

**- Print "Invalid Input."**

**Step 4: Insert Element (insert\_element() function)**

**- Ask user to enter the number of elements for array1 and array2**

**- Insert elements into array1 and array2**

**- Print array1 and array2**

**Step 5: Merge Element (merge\_element() function)**

**- Merge array1 and array2 into array3**

**- Print the merged array (array3)**

**Step 6: Sort Element (sort\_element() function)**

**- Sort array3 in ascending order**

**- Print the sorted array (array3)**

**Step 7: Display Element (display\_element() function)**

**- Print array1, array2, and the merged and sorted array (array3)**

**Step 8: Exit**

**- Exit the menu**

**- Print "Harshit Jaiswal 22" for student identity.**

***Code:***

**array1 = []**

**array2 = []**

**array3 = []**

**print("\nMenu")**

**print("1. Insertion Element")**

**print("2. Merge Element")**

**print("3. Sort Element")**

**print("4. Display Element")**

**print("5. Exit")**

**def insert\_element():**

**global array1, array2**

**num1 = int(input("Enter the number of elements for array1 and array2 : "))**

**print("Inserting elements in Array 1.")**

**for i in range(0, num1):**

**num2 = int(input("Enter element {} for array 1 : ".format(i+1)))**

**array1.append(num2)**

**print("Inserting elements in Array 2.")**

**for j in range(0, num1):**

**num3 = int(input("Enter element {} for array 2 : ".format(j+1)))**

**array2.append(num3)**

**print("Array 1:", array1)**

**print("Array 2:", array2)**

**def merge\_element():**

**global array3**

**array3 = array1 + array2**

**print("Merged array :", array3)**

**def sort\_element():**

**array3.sort()**

**print("Sorted array :", array3)**

**def display\_element():**

**print("Array 1:", array1)**

**print("Array 2:", array2)**

**print("Merged and Sorted array:", array3)**

**while True:**

**ch = int(input("Enter your choice: "))**

**if ch == 1:**

**insert\_element()**

**elif ch == 2:**

**merge\_element()**

**elif ch == 3:**

**sort\_element()**

**elif ch == 4:**

**display\_element()**

**elif ch == 5:**

**print("Exiting Menu")**

**break**

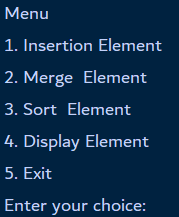
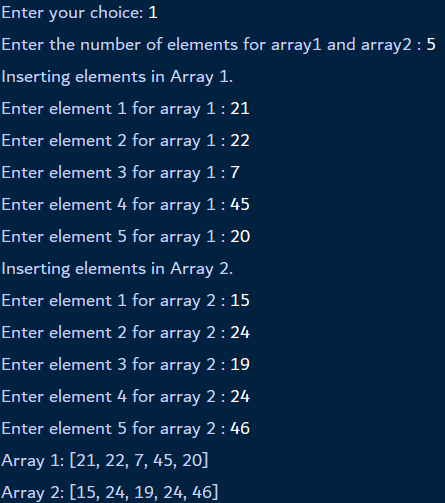
**else:**

**print("Invalid Input.")**

**print("Harshit Jaiswal 22")**

***Output:***

**1. Menu 2. Insertion Of Element**

****

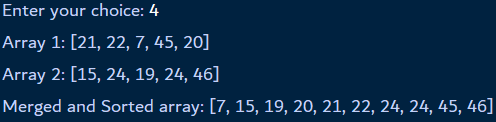
**3. Merge Element**

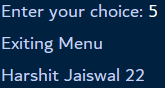
****

**4. Sort Element**

****

**5. Display Element**

****

**6. Exit menu**

**3. Write a program to perform the Matrix addition, Multiplication and**

**Transpose Operation. [Menu Driven]**

***Algorithm:***

**Step 1: Print Matrix 1**

**- Print "Matrix 1 :"**

**- Iterate through each row in matrix1**

**- Iterate through each element in the current row**

**- Print the element followed by a space (using end="")**

**Step 2: Print Matrix 2**

**- Print "Matrix 2 :"**

**- Iterate through each row in matrix2**

**- Iterate through each element in the current row**

**- Print the element followed by a space (using end="")**

**Step 3: Initialize Result Matrix**

**- Create a result matrix result with the same dimensions as matrix1 (2x2 in this case)**

**- Initialize all elements of result to 0**

**Step 4: Perform Matrix Addition**

**- Iterate through each row i in matrix1**

**- Iterate through each column j in matrix1 (which is also the column index for matrix2)**

**- Add the corresponding elements of matrix1 and matrix2 and store the result in result[i][j]**

**Step 5: Print Result Matrix**

**- Print "Addition :"**

**- Iterate through each row in result**

**- Iterate through each element in the current row**

**- Print the element followed by a space (using end="")**

**- Print "Harshit Jaiswal 22" for student identity.**

***Code:***

**For Addition:**

**matrix1 = [[1,2],[3,4]]**

**matrix2 = [[5,6],[7,8]]**

**print("Matrix 1 is :")**

**for row in matrix1:**

**for element in row:**

**print(element, end=" ")**

**print()**

**print("Matrix 2 is :")**

**for row in matrix2:**

**for element in row:**

**print(element, end=" ")**

**print()**

**print ("Addition of Matrix 1 and Matrix 2")**

**result = [[0,0],[0,0]]**

**for i in range(len(matrix1)):**

**for j in range(len(matrix1[0])):**

**result[i][j] = matrix1[i][j] + matrix2[i][j]**

**for row in result:**

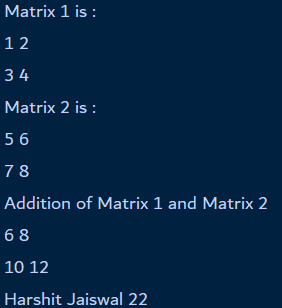
**for element in row:**

**print(element, end=" ")**

**print()**

**print("Harshit Jaiswal 22")**

***Output*:**

****

**For Multiplication:**

***Algorithm*:**

**Step 1: Print Matrix 1**

**- Iterate through each row in matrix1**

**- Iterate through each element in the current row**

**- Print the element followed by a space (using end=" ")**

**Step 2: Print Matrix 2**

**- Iterate through each row in matrix2**

**- Iterate through each element in the current row**

**- Print the element followed by a space (using end=" ")**

**Step 3: Initialize Result Matrix**

**- Create a result matrix result with the same dimensions as matrix1 (2x2 in this case)**

**- Initialize all elements of result to 0**

**Step 4: Perform Matrix Multiplication**

**- Iterate through each row i in matrix1**

**- Iterate through each column j in matrix1 (which is also the column index for matrix2)**

**- Iterate through each element k in the current row of matrix1 (which is also the row index for matrix2)**

**- Calculate the dot product of the current row of matrix1 and the current column of matrix2 and add it to the corresponding element in result**

**- Store the result in result[i][j]**

**Step 5: Print Result Matrix**

**- Iterate through each row in result**

**- Iterate through each element in the current row**

**- Print the element followed by a space (using end=" ")**

**- Print "Harshit Jaiswal 22" for student identity.**

***Code:***

**matrix1 = [[1,2],[3,4]]**

**matrix2 = [[5,6],[7,8]]**

**print("Matrix 1 is :")**

**for row in matrix1:**

**for element in row:**

**print(element, end=" ")**

**print()**

**print("Matrix 2 is :")**

**for row in matrix2:**

**for element in row:**

**print(element, end=" ")**

**print()**

**print ("Multiplication of Matrix 1 and Matrix 2")**

**result\_mul = [[0,0],[0,0]]**

**for i in range(len(matrix1)):**

**for j in range(len(matrix2[0])):**

**for k in range(len(matrix2)):**

**result\_mul[i][j] += matrix1[i][k] \* matrix2[k][j]**

**for row in result\_mul:**

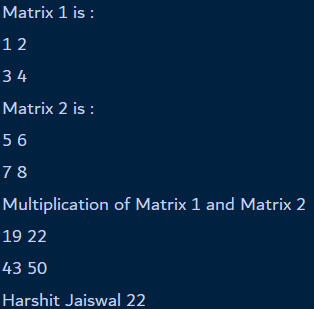
**for element in row:**

**print(element, end=" ")**

**print()**

**print("Harshit Jaiswal 22")**

***Output*:**

****

**For Transpose:**

***Algorithm:***

**Step 1: Print Matrix 2**

**- Print "Matrix 2 :"**

**- Iterate through each row in matrix2**

**- Iterate through each element in the current row**

**- Print the element followed by a space (using end="")**

**Step 2: Initialize Result Matrix**

**- Create a result matrix result with the same dimensions as matrix2 (2x2 in this case)**

**- Initialize all elements of result to 0**

**Step 3: Perform Matrix Transpose**

**- Iterate through each row i in matrix2**

**- Iterate through each column j in matrix2**

**- Assign the element at matrix2[j][i] to result[i][j] (swapping row and column indices)**

**Step 4: Print Transposed Matrix**

**- Print "Transpose :"**

**- Iterate through each row in result**

**- Iterate through each element in the current row**

**- Print the element followed by a space (using end="")**

**- Print "Harshit Jaiswal 22" for student identity.**

***Code:***

**matrix2 = [[5,6],[7,8]]**

**print("Matrix is :")**

**for row in matrix2:**

**for element in row:**

**print(element, end=" ")**

**print()**

**print ("Transpose of Matrix ")**

**result\_trans2 = [[0,0],[0,0]]**

**for i in range(len(matrix2)):**

**for j in range(len(matrix2[0])):**

**result\_trans2[j][i] = matrix2[i][j]**

**for row in result\_trans2:**

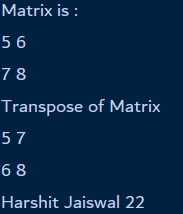
**for element in row:**

**print(element, end=" ")**

**print()**

**print("Harshit Jaiswal 22")**

***Output:***

****

**Laxmi Charitable Trust’s**

**Sheth L.U.J College of Arts & Sir M.V. College of Science and Commerce**

Department of Information Technology (Bsc.IT Semester IV)

Data Structures Practical

Practical - II

|  |  |
| --- | --- |
| Roll No.: S022 | Name: Harshit Jaiswal |
| Class: SYIT | Batch: 01 |
| Date of Assignment: 31-07-2024 | Date/Time of Submission: |

**2. Implement the following for Linked List:**

**a. Write a program to create a single linked list and display the node elements in reverse order.**

***Algorithm:***

**Step 1: Create an empty list to store node data**

**- Initialize an empty list nodes\_data to store the data of nodes**

**Step 2: Traverse the linked list and store node data**

**- Initialize a pointer current to the head of the linked list (node1)**

**- While current is not None:**

**1. Append the data of the current node to nodes\_data: nodes\_data.append(current.data)**

**2. Move current to the next node: current = current.next**

**Step 3: Print the linked list in reverse order**

**- Use the reversed function to reverse the nodes\_data list**

**- Iterate through the reversed list and print each data element followed by " - > "**

**- Print "None" at the end to indicate the end of the linked list**

**- Print "Harshit Jaiswal 22" for student identity.**

***Code:***

**class Node:**

**def \_\_init\_\_(self, data):**

**self.data = data**

**self.next = None**

**# Creating Nodes**

**node1 = Node(1)**

**node2 = Node(2)**

**node3 = Node(3)**

**node4 = Node(4)**

**# Connecting the Nodes**

**node1.next = node2**

**node2.next = node3**

**node3.next = node4**

**# Using a list to store the data of nodes**

**nodes\_data = []**

**current = node1**

**while current is not None:**

**nodes\_data.append(current.data)**

**current = current.next**

**# Printing the linked list in reverse order**

**for data in reversed(nodes\_data):**

**print(data, end = " - > ")**

**print("None")**

**print("Harshit Jaiswal 22")**

***Output:***

****

**b. Write a program to search the elements in the linked list and display the same**

***Algorithm:***

**Step 1: Create an empty list to store node data**

**- Initialize an empty list nodes\_data to store the data of nodes**

**Step 2: Traverse the linked list and store node data**

**- Initialize a pointer current to the head of the linked list (node1)**

**- While current is not None:**

**1. Append the data of the current node to nodes\_data: nodes\_data.append(current.data)**

**2. Move current to the next node: current = current.next**

**Step 3: Print the linked list in reverse order**

**- Use the reversed function to reverse the nodes\_data list**

**- Iterate through the reversed list and print each data element followed by " - > "**

**- Print "None" at the end to indicate the end of the linked list**

**Step 4: Search for a data element**

**- Take an integer input a from the user**

**- Check if a exists in nodes\_data using the in operator**

**- If a exists, print "a exists"**

**- If a does not exist, print "a does not exist"**

**- Print "Harshit Jaiswal 22" for student identity.**

***Code:***

**class Node:**

**def \_\_init\_\_(self, data):**

**self.data = data**

**self.next = None**

**# Creating Nodes**

**node1 = Node(1)**

**node2 = Node(2)**

**node3 = Node(3)**

**node4 = Node(4)**

**# Connecting the Nodes**

**node1.next = node2**

**node2.next = node3**

**node3.next = node4**

**# Using a list to store the data of nodes**

**nodes\_data = []**

**current = node1**

**while current is not None:**

**nodes\_data.append(current.data)**

**current = current.next**

**# Printing the linked list in reverse order**

**for data in reversed(nodes\_data):**

**print(data, end = " - > ")**

**print("None")**

**a = int(input("Enter the data : "))**

**if a in nodes\_data:**

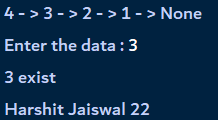
**print(a, "exist")**

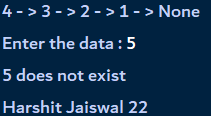
**else:**

**print(a, "does not exist")**

**print("Harshit Jaiswal 22")**

***Output:***





**c. Write a program to create double linked list and sort the elements in the linked list.**

***Algorithm:***

**-Node Creation:**

**Step 1: Create a Node instance node1 with data 10.**

**Step 2: Create a Node instance node2 with data 20.**

**Step 3: Create a Node instance node3 with data 30.**

**Step 4: Create a Node instance node4 with data 40.**

**-Node Insertion:**

**Step 5: Set the next pointer of node1 to node2.**

**Step 6: Set the prev pointer of node1 to None (as it's the head node).**

**Step 7: Set the next pointer of node2 to node3.**

**Step 8: Set the prev pointer of node2 to node1.**

**Step 9: Set the next pointer of node3 to node4.**

**Step 10: Set the prev pointer of node3 to node2.**

**Step 11: Set the next pointer of node4 to None (as it's the tail node).**

**Step 12: Set the prev pointer of node4 to node3.**

**-Print the List in Reverse:**

**Step 13: Initialize current to node4 (the tail node).**

**Step 14: Traverse the list backward from node4 to node1 using the prev pointers:**

**Print the data of current.**

**Move to the previous node by updating current to current.prev.**

**Step 15: Print "None" to signify the end of the list.**

**Step 16: Print "Harshit Jaiswal 22" to indicate the end of the output.**

***Code:***

**class Node:**

**def \_\_init\_\_(self,data):**

**self.data = data**

**self.next = None**

**self.prev = None**

**#node Creation**

**node1 = Node(10)**

**node2 = Node(20)**

**node3 = Node(30)**

**node4 = Node(40)**

**#node Insertion**

**node1.next = node2**

**node1.prev = None**

**node2.next = node3**

**node2.prev = node1**

**node3.next = node4**

**node3.prev = node2**

**node4.next = None**

**node4.prev = node3**

**#Print**

**current = node4**

**while current is not None:**

**print(current.data,end=" -> ")**

**current = current.prev**

**print("None")**

**print("Harshit Jaiswal 22")**

***Output:***

**Laxmi Charitable Trust’s**

**Sheth L.U.J College of Arts & Sir M.V. College of Science and Commerce**

Department of Information Technology (Bsc.IT Semester III)

**Data Structures Practical**

Practical - III

|  |  |
| --- | --- |
| Roll No.:22 | Name: Harshit Jaiswal |
| Class: SYIT | Batch: B1 |
| Date of Assignment: | Date/Time of Submission: |

**3. Implement the following for Stack:**

**a. Write a program to implement the concept of Stack with Push, Pop, Display**

**and Exit operations.**

***Algorithm*:**

**Step 1: Display Menu**

**- Print the menu options:**

**1. Push**

**2. Pop**

**3. Display**

**4. Exit**

**Step 2: Get User Choice**

**- Ask the user to enter their choice (1-4)**

**- Try to convert the input to an integer**

**- If invalid input, print an error message and repeat Step 1**

**Step 3: Perform Action**

**- If choice is 1 (Push):**

**- Ask the user to enter a value to push**

**- Try to push the value onto the stack**

**- If stack is full, print an error message**

**- Display the updated stack contents**

**- If choice is 2 (Pop):**

**- Try to pop an item from the stack**

**- If stack is empty, print an error message**

**- Display the updated stack contents**

**- If choice is 3 (Display):**

**- Display the current stack contents**

**- If choice is 4 (Exit):**

**- Print a goodbye message and exit the program**

**Step 4: Repeat**

**- Repeat Steps 1-3 until the user chooses to exit (choice 4)**

**Stack Operations**

**- Push:**

**1. Check if the stack is full**

**2. If not full, add the new value to the top of the stack**

**3. Print a success message**

**- Pop:**

**1. Check if the stack is empty**

**2. If not empty, remove the top item from the stack**

**3. Print a success message**

**- Display:**

**1. Check if the stack is empty**

**2. If not empty, print the current stack contents**

**print(“Harshit Jaiswal 22”) as a student identity.**

**Code:**

**def menu():**

**print("\nMenu")**

**print("1. Push")**

**print("2. Pop")**

**print("3. Display")**

**print("4. Exit")**

**class Stack:**

**def \_\_init\_\_(self, max\_size=5):**

**self.values = []**

**self.max\_size = max\_size**

**def push(self, x):**

**if len(self.values) < self.max\_size:**

**self.values.insert(0, x)**

**print(f"Item {x} pushed to stack.")**

**else:**

**print(f"Stack is full. Cannot push {x}.")**

**self.display()**

**def pop(self):**

**if not self.is\_empty():**

**item = self.values.pop(0)**

**print(f"Item {item} popped from stack.")**

**else:**

**print("Stack is empty. Cannot pop.")**

**self.display()**

**def display(self):**

**if self.values:**

**print("Stack contents:", self.values)**

**else:**

**print("Stack is empty.")**

**def is\_empty(self):**

**return len(self.values) == 0**

**s = Stack()**

**while True:**

**menu()**

**try:**

**ch = int(input("Enter your choice: "))**

**if ch == 1:**

**value = int(input("Enter value to push: "))**

**s.push(value)**

**elif ch == 2:**

**s.pop()**

**elif ch == 3:**

**s.display()**

**elif ch == 4:**

**print("Exiting...")**

**break**

**else:**

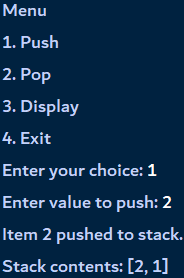
**print("Invalid choice. Please try again.")**

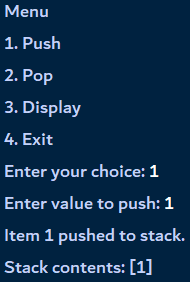
**except ValueError:**

**print("Invalid input. Please enter a numeric value.")**

**print("Harshit Jaiswal 22")**

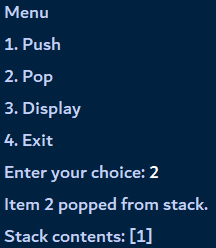
**Output:**

**1. Stack**

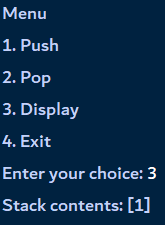
****

**2. P**

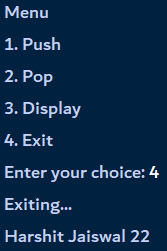
**2. Pop**

****

**3. Display**

****

**4. Exit**

****

**b. Write a program to implement the Tower of Hanoi problem.**

***Algorithm*:**

**Step 1: Base Case**

**- If there is only 1 disk, move it directly from the source peg to the target peg.**

**Step 2: Recursive Case**

**- Move n-1 disks from the source peg to the auxiliary peg, using the target peg as a temporary storage.**

**- Call the Tower of Hanoi function recursively with:**

**- Number of disks: n-1**

**- Source peg: current source peg**

**- Auxiliary peg: current target peg**

**- Target peg: current auxiliary peg**

**Step 3: Move the nth Disk**

**- Move the nth disk from the source peg to the target peg.**

**Step 4: Recursive Case (again)**

**- Move the n-1 disks from the auxiliary peg to the target peg, using the source peg as a temporary storage.**

**- Call the Tower of Hanoi function recursively with:**

**- Number of disks: n-1**

**- Source peg: current auxiliary peg**

**- Auxiliary peg: current source peg**

**- Target peg: current target peg**

**Step 5: Print the Move**

**- Print the move made in Step 3, indicating the disk number and the pegs involved.**

**Step 6: Repeat**

**- Repeat Steps 2-5 until all disks are moved from the source peg to the target peg.**

**Code:**

**def toh (disks, source, aux, target):**

**if(disks == 1):**

**print("Move disc 1 from {} to {} . ".format(source, target))**

**return**

**toh(disks - 1, source, target, aux)**

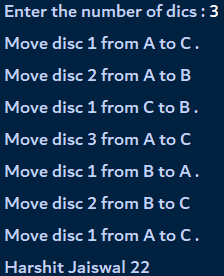
**print("Move disc {} from {} to {}". format(disks, source, target))**

**toh(disks - 1, aux, source, target)**

**disks = int(input("Enter the number of dics : "))**

**toh(disks, 'A','B','C')**

**print("Harshit Jaiswal 22")**

**Output:**

**Laxmi Charitable Trust’s**

**Sheth L.U.J College of Arts & Sir M.V. College of Science and Commerce**

Department of Information Technology (Bsc.IT Semester IV)

Data Structure Practical

Practical - IV

|  |  |
| --- | --- |
| Roll No.: 22 | Name: Harshit Jaiswal |
| Class: SYIT | Batch: 01 |
| Date of Assignment: 21-08-24 | Date/Time of Submission: 21-08-24 |

**4. Implement the following for Queue:**

**a. Write a program to implement the concept of Queue with Insert, Delete, Display and Exit operations.**

**Algorithm:**

**1. Initialization:**

**- Create a Queue class with an empty list queue.**

**2. Menu Display:**

**- Display a menu with options to Insert, Delete, Display, and Exit.**

**3. User Input Loop:**

**- Continuously prompt the user to enter an operation number until they choose to Exit.**

**4. Insert Operation:**

**- If user chooses Insert (option 1):**

**- Ask for the number of elements to insert.**

**- For each element, ask for the value and enqueue it.**

**- Print a success message indicating the number of elements inserted.**

**5. Delete Operation:**

**- If user chooses Delete (option 2):**

**- Dequeue an element.**

**- If the queue is not empty, print the removed element.**

**6. Display Operation:**

**- If user chooses Display (option 3):**

**- If the queue is empty, print "Queue is empty."**

**- Otherwise, print the current queue elements.**

**7. Exit Operation:**

**- If user chooses Exit (option 4):**

**- Print "Exiting..." and break out of the loop.**

**8. Invalid Choice:**

**- If user enters an invalid choice, print an error message and prompt again.**

**- Print “Harshit Jaiswal 22” as student identity.**

**Code:**

**class Queue:**

**def \_\_init\_\_(self):**

**self.queue = []**

**def enqueue(self, item):**

**self.queue.append(item)**

**def dequeue(self):**

**if len(self.queue) < 1:**

**print("Queue is empty, cannot dequeue.")**

**return None**

**return self.queue.pop(0)**

**def display(self):**

**if len(self.queue) < 1:**

**print("Queue is empty.")**

**else:**

**print("Queue:", self.queue)**

**def size(self):**

**return len(self.queue)**

**def menu():**

**print("MENU")**

**print("1. Insert")**

**print("2. Delete")**

**print("3. Display")**

**print("4. Exit")**

**q = Queue()**

**menu()**

**while True:**

**ch = int(input("Enter the operation number : "))**

**if ch == 1:**

**num\_elements = int(input("Enter the number of element to insert : "))**

**for \_ in range(num\_elements):**

**item = int(input("Enter the element to insert : "))**

**q.enqueue(item)**

**print(f"{num\_elements} elements inserted.")**

**elif ch == 2:**

**removed\_item = q.dequeue()**

**if removed\_item is not None:**

**print(f"{removed\_item} removed.")**

**elif ch == 3:**

**q.display()**

**elif ch == 4:**

**print("Exiting...")**

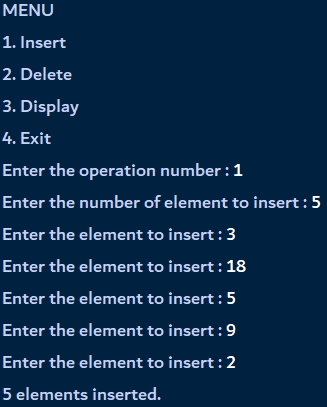
**break**

**else:**

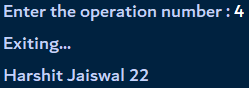
**print("Invalid choice. Please try again.")**

**print("Harshit Jaiswal 22")**

**Display:**

**1. Inserting 2. Deleting**

****

**3. Display 4. Exit**

**b. Write a program to implement the concept of Circular Queue**

**Algorithm:**

**1. Initialization:**

**- Create an empty list items to store elements.**

**- Initialize the Deque class with an empty list.**

**2. Check if Deque is Empty:**

**- Return True if the items list is empty, False otherwise.**

**3. Add Element to Rear:**

**- Append an element item to the end of the items list using append().**

**- Repeat step 3 for each element to be added to the rear (7, 15, 22, 25 in this case).**

**4. Add Element to Front:**

**- Insert an element item at the beginning of the items list using insert(0, item).**

**- Repeat step 4 for each element to be added to the front (20, 21, 24 in this case).**

**5. Remove Element from Front:**

**- Remove and return the front element of the items list using pop(0).**

**- Repeat step 5 to remove an element from the front (21 in this case).**

**6. Remove Element from Rear:**

**- Remove and return the rear element of the items list using pop().**

**- Repeat step 6 to remove an element from the rear (22 in this case).**

**7. Get Deque Size:**

**- Return the number of elements in the items list using len().**

**8. Display Deque Elements:**

**- Print the current state of the items list.**

**- Print “Harshit Jaiswal 22” as student identity.**

**Code:**

**class deque:**

**def \_\_init\_\_(self):**

**self.items = []**

**def isEmpty(self):**

**return self.items == []**

**def addRear(self, item):**

**self.items.append(item)**

**def addFront(self, item):**

**self.items.insert(0, item)**

**def removeFront(self):**

**return self.items.pop(0)**

**def removeRear(self):**

**return self.items.pop()**

**def size(self):**

**return len(self.items)**

**d = deque()**

**print(d.isEmpty())**

**d.addRear(7)**

**d.addRear(15)**

**d.addFront(20)**

**d.addFront(21)**

**print(d.size())**

**print(d.isEmpty())**

**d.addRear(22)**

**print(d.removeRear())**

**print(d.removeFront())**

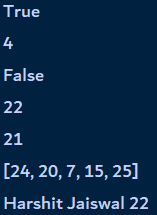
**d.addFront(24)**

**d.addRear(25)**

**print(d.items)**

**print("Harshit Jaiswal 22")**

**Output:**

****

**c. Write a program to implement the concept of Deque.**

**Algorithm:**

**1. Initialization:**

**- Create a MyCircularQueue class with a fixed size k.**

**- Initialize an empty queue with k slots, all set to None.**

**- Set head and tail pointers to -1, indicating an empty queue.**

**2. Enqueue Operation:**

**- Check if the queue is full by comparing tail + 1 (mod k) with head.**

**- If full, print "The circular queue is full".**

**- If empty, set head and tail to 0 and add the element at tail.**

**- Otherwise, increment tail (mod k) and add the element at tail.**

**3. Dequeue Operation:**

**- Check if the queue is empty by checking if head is -1.**

**- If empty, print "The circular queue is empty".**

**- If only one element, remove it and reset head and tail to -1.**

**- Otherwise, increment head (mod k) and remove the element at head.**

**4. Print Circular Queue:**

**- Check if the queue is empty by checking if head is -1.**

**- If empty, print "No element in the circular queue".**

**- If tail is greater than or equal to head, print elements from head to tail.**

**- Otherwise, print elements from head to the end of the queue, then from the beginning to tail.**

**5. Create and Test the Circular Queue:**

**- Create a MyCircularQueue object with size 5.**

**- Enqueue elements 10, 20, 30, 40, and 50.**

**- Print the initial queue.**

**- Dequeue an element.**

**- Print the queue after removing an element.**

**Code:**

**class MyCircularQueue():**

**def \_\_init\_\_(self, k):**

**self.k = k**

**self.queue = [None] \* k**

**self.head = self.tail = -1**

**# Insert an element into the circular queue**

**def enqueue(self, data):**

**if ((self.tail + 1) % self.k == self.head):**

**print("The circular queue is full\n")**

**elif (self.head == -1):**

**self.head = 0**

**self.tail = 0**

**self.queue[self.tail] = data**

**else:**

**self.tail = (self.tail + 1) % self.k**

**self.queue[self.tail] = data**

**# Delete an element from the circular queue**

**def dequeue(self):**

**if (self.head == -1):**

**print("The circular queue is empty\n")**

**elif (self.head == self.tail):**

**temp = self.queue[self.head]**

**self.head = -1**

**self.tail = -1**

**return temp**

**else:**

**temp = self.queue[self.head]**

**self.head = (self.head + 1) % self.k**

**return temp**

**def printCQueue(self):**

**if(self.head == -1):**

**print("No element in the circular queue")**

**elif (self.tail >= self.head):**

**for i in range(self.head, self.tail + 1):**

**print(self.queue[i], end=" ")**

**print()**

**else:**

**for i in range(self.head, self.k):**

**print(self.queue[i], end=" ")**

**for i in range(0, self.tail + 1):**

**print(self.queue[i], end=" ")**

**print()**

**obj = MyCircularQueue(5)**

**obj.enqueue(10)**

**obj.enqueue(20)**

**obj.enqueue(30)**

**obj.enqueue(40)**

**obj.enqueue(50)**

**print("Initial queue")**

**obj.printCQueue()**

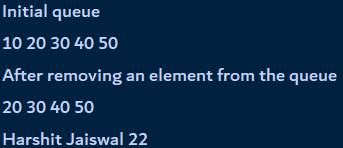
**obj.dequeue()**

**print("After removing an element from the queue")**

**obj.printCQueue()**

**print("Harshit Jaiswal 22")**

**Output:**

****

**Laxmi Charitable Trust’s**

**Sheth L.U.J College of Arts & Sir M.V. College of Science and Commerce**

Department of Information Technology (Bsc.IT Semester IV)

Data Structures Practical

Practical - V

|  |  |
| --- | --- |
| Roll No.: S022 | Name: Harshit Jaiswal |
| Class: SYIT | Batch: 01 |
| Date of Assignment: 28-08-2024 | Date/Time of Submission: 28- 8-2024 |

**Implement the following sorting techniques:**

1. **Write a program to implement bubble sort.**

**ALGORITHM:**

**Step 1: Start**

**Step 2: Define a function to sort the list.**

**Step 3: Go through the list multiple times.**

**Step 4: During each pass, compare each item with the next one.**

**Step 5: If an item is bigger than the next one, swap them.**

**Step 6: If no swaps happen in a pass, the list is sorted, so stop.**

**Step 7: Use the function to sort the list [-2, 45, 0, 11, -9].**

**Step 8: Print "Sorted Array in Ascending Order:" followed by the sorted list.**

**Step 9: Print "Harshit Jaiswal 22".**

**Step 10: End**

**CODE:**

**def bubbleSort(array):**

**for i in range(len(array)):**

**for j in range(0, len(array) - i - 1):**

**if array[j] > array[j + 1]:**

**temp = array[j]**

**array[j] = array[j+1]**

**array[j+1] = temp**

**data = [-2, 45, 0, 11, -9]**

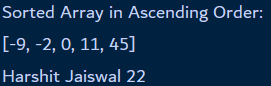
**bubbleSort(data)**

**print('Sorted Array in Ascending Order:')**

**print(data)**

**print("Harshit Jaiswal 22")**

**OUTPUT:**

****

1. **Write a program to implement selection sort.**

**ALGORITHM:**

**Step 1: Start**

**Step 2: Define a function to sort the list.**

**Step 3: Go through the list one item at a time.**

**Step 4: For each item, find the smallest item in the rest of the list.**

**Step 5: Swap the current item with the smallest item found.**

**Step 6: Repeat until the whole list is sorted.**

**Step 7: Use the function to sort the list [-26, 5, 0, 1, -69].**

**Step 8: Print "Sorted Array in Ascending Order:" followed by the sorted list.**

**Step 9: Print "Harshit Jaiswal 22".**

**Step 10: End**

**CODE:**

**def selectionSort(array, size):**

**for step in range(size):**

**min\_idx = step**

**for i in range(step + 1, size):**

**if array[i] < array[min\_idx]:**

**min\_idx = i**

**(array[step], array[min\_idx]) = (array[min\_idx], array[step])**

**data = [22,1,15,84,17,10,55,5]**

**size = len(data)**

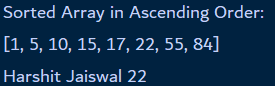
**selectionSort(data, size)**

**print('Sorted Array in Ascending Order:')**

**print(data)**

**print("Harshit Jaiswal 22")**

**OUTPUT:**

****

1. **Write a program to implement insertion sort.**

**Algorithm:**

**Step 1: Start**

**Step 2: Define a function to sort the list.**

**Step 3: Begin with the second item in the list.**

**Step 4: Compare this item with the items before it.**

**Step 5: Move the larger items one position to the right.**

**Step 6: Place the current item in its correct position.**

**Step 7: Repeat this process for each item in the list until the entire list is sorted.**

**Step 8: Use the function to sort the list [9, 5, 1, 4, 3].**

**Step 9: Print "Sorted Array in Ascending Order:" followed by the sorted list.**

**Step 10: Print " Harshit Jaiswal 22".**

**Step 11: End**

**CODE:**

**def insertionSort(array):**

**for step in range(1, len(array)):**

**key = array[step]**

**j = step - 1**

**while j >= 0 and key < array[j]:**

**array[j + 1] = array[j]**

**j = j - 1**

**array[j + 1] = key**

**data = [9, 5, 1, 4, 3]**

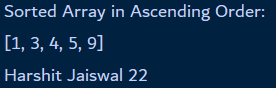
**insertionSort(data)**

**print('Sorted Array in Ascending Order:')**

**print(data)**

**print("Harshit Jaiswal 22")**

**OUTPUT:**

****

**Laxmi Charitable Trust’s**

**Sheth L.U.J College of Arts & Sir M.V. College of Science and Commerce**

Department of Information Technology (Bsc.IT Semester IV)

Data Structures Practical

Practical - VI

|  |  |
| --- | --- |
| Roll No.: S022 | Name: Harshit Jaiswal |
| Class: SYIT | Batch: 01 |
| Date of Assignment: 04-09-2024 | Date/Time of Submission: |

**1. Write a program to implement merge sort.**

**Algorithm:**

**Step 1: Divide**

**- If the length of the array is greater than 1, divide the array into two halves:**

**- Left half (L): elements from index 0 to r-1**

**- Right half (M): elements from index r to n-1**

**- where r = len(array)//2**

**Step 2: Recursively Sort**

**- Recursively call mergeSort on the two halves:**

**- mergeSort(L)**

**- mergeSort(M)**

**Step 3: Merge**

**- Merge the two sorted halves, L and M, into a single sorted array:**

**1. Initialize three pointers:**

**- i: index for L**

**- j: index for M**

**- k: index for the original array**

**2. Compare elements from L and M:**

**- If L[i] < M[j], copy L[i] to array[k]**

**- Else, copy M[j] to array[k]**

**3. Increment the corresponding pointer (i or j) and k**

**4. Repeat until one of the halves is exhausted**

**Step 4: Copy Remaining Elements**

**- If there are remaining elements in either L or M:**

**1. Copy the remaining elements from L to the end of the original array**

**2. Copy the remaining elements from M to the end of the original array**

**Step 5: Repeat**

**- Repeat the process until the entire array is sorted**

**Step 6: Print Sorted Array**

**- Print the sorted array using the printList function**

**- Print “Harshit Jaiswal 22” as a student identity.**

**Code:**

**def mergeSort(array):**

**if len(array) > 1:**

**r = len(array)//2**

**L = array[:r]**

**M = array[r:]**

**mergeSort(L)**

**mergeSort(M)**

**i = j = k = 0**

**while i < len(L) and j < len(M):**

**if L[i] < M[j]:**

**array[k] = L[i]**

**i += 1**

**else:**

**array[k] = M[j]**

**j += 1**

**k += 1**

**while i < len(L):**

**array[k] = L[i]**

**i += 1**

**k += 1**

**while j < len(M):**

**array[k] = M[j]**

**j += 1**

**k += 1**

**def printList(array):**

**for i in range(len(array)):**

**print(array[i], end=" ")**

**print()**

**if \_\_name\_\_ == '\_\_main\_\_':**

**array = [6, 5, 12, 10, 9, 1]**

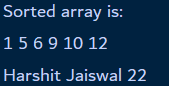
**mergeSort(array)**

**print("Sorted array is: ")**

**printList(array)**

**print("Harshit Jaiswal 22")**

**Output:**

****

**2. Write a program to search the element using sequential search.**

**Algorithm:**

**Step 1: Initialize**

**- Set the search element x**

**- Set the array array**

**- Get the length of the array n using len(array)**

**Step 2: Iterate and Compare**

**- Start at the first element of the array (index i = 0)**

**- Compare the current element array[i] with the search element x**

**- If array[i] == x, go to Step 3**

**- Else, increment i and repeat Step 2**

**Step 3: Found**

**- Return the index i where the match was found**

**Step 4: Not Found**

**- If i reaches the end of the array (i == n), return -1 to indicate that the element is not found**

**Step 5: Print Result**

**- Print the search element x**

**- If the result is -1, print "Element not found"**

**- Else, print "Element found at index: " followed by the result**

**- Print “Harshit Jaiswal 22” as a student identity.**

**Code:**

**def linearSearch(array, n, x):**

**for i in range(0, n):**

**if (array[i] == x):**

**return i**

**return -1**

**array = [2, 4, 0, 1, 9]**

**x = 1**

**n = len(array)**

**result = linearSearch(array, n, x)**

**print("Element to find :",x)**

**if(result == -1):**

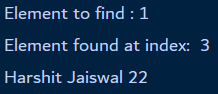
**print("Element not found")**

**else:**

**print("Element found at index: ", result)**

**print("Harshit Jaiswal 22")**

**Output:**

****

**3. Write a program to search the element using sequential search.**

**Algorithm:**

**Step 1: Initialize**

**- Set the search element x**

**- Set the array array**

**- Set the low index low to 0**

**- Set the high index high to the last index of the array (len(array) - 1)**

**Step 2: Check Bounds**

**- If high is greater than or equal to low, proceed to Step 3**

**- Else, return -1 (element not found)**

**Step 3: Calculate Mid**

**- Calculate the mid index mid using the formula: low + (high - low) // 2**

**Step 4: Compare Mid Element**

**- Compare the mid element array[mid] with the search element x**

**- If x matches array[mid], return mid (element found)**

**- If x is greater than array[mid], proceed to Step 5**

**- If x is less than array[mid], proceed to Step 6**

**Step 5: Search Right Half**

**- Recursively call binarySearch with:**

**- array unchanged**

**- x unchanged**

**- low set to mid + 1**

**- high unchanged**

**Step 6: Search Left Half**

**- Recursively call binarySearch with:**

**- array unchanged**

**- x unchanged**

**- low unchanged**

**- high set to mid - 1**

**Step 7: Print Result**

**- Print the search element x**

**- If the result is not -1, print "Element is present at index" followed by the result**

**- Else, print "Not found"**

**- Print "Harshit Jaiswal 22" for student identity.**

**Code:**

**def binarySearch(array, x, low, high):**

**if high >= low:**

**mid = low + (high - low)//2**

**if x == array[mid]:**

**return mid**

**elif x > array[mid]:**

**return binarySearch(array, x, mid + 1, high)**

**else:**

**return binarySearch(array, x, low, mid - 1)**

**else:**

**return -1**

**array = [3, 4, 5, 6, 7, 8, 9]**

**x = 4**

**result = binarySearch(array, x, 0, len(array)-1)**

**print("Element to find :", x)**

**if result != -1:**

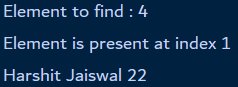
**print("Element is present at index " + str(result))**

**else:**

**print("Not found")**

**print("Harshit Jaiswal 22")**

**Output:**

****