Bahria University,

Karachi Campus



COURSE: CSC-221 DATA STRUCTURES AND ALGORITHM

TERM: FALL 2020, CLASS: BSE- 3 (A)

Submitted By:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ADIL WAHEED) (65190)

Submitted To:

Engr. Maam Nazar Mobeen/ Engr. Ramshaa

Signed Remarks: Score:

INDEX

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SNO | DATE | LAB NO | LAB OBJECTIVE | SIGN |
| 01 | 1-10-2020 | 01 | ONE AND TWO DIMENSIONAL ARRAY |  |
| 02 | 09-10-20 | 02 | Linear Search & Sorting Algorithms |  |
| 03 | 13-10-20 | 03 | Recusrion |  |
| 04 | 30/10/2020 | 04 | Binary Search Algorithm |  |
| 05 | 30/10/2020 | 05 | Merge Sort |  |
| 06 | 30/10/2020 | 06 | Quick Sort |  |
| 07 | 4/11/2020 | 07 | Stack |  |
| 08 | 12/11/2020 | 08 | QUEUE |  |
| 09 | 20/10/2020 | 09 | Doubly Linked List |  |
| 10 | 24/10/2020 | 10 | CIRCULAR Linked List |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| SNO | DATE | LAB NO | LAB OBJECTIVE | SIGN |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Bahria University,

Karachi Campus



LAB EXPERIMENT NO.

\_\_\_10\_\_\_\_

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| 1 | **Write a program to create a circular linked list and apply sorting algorithm.** |
| 2 | **Implement dynamic queue using circular linked list.** |
|  |  |
|  |  |
|  |  |

Submitted On:

\_\_\_\_\_\_\_\_\_\_\_\_

(Date: 24/12/20)

**Task No. 1: Write a program to create a circular linked list and apply sorting algorithm.**

**Solution:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace newcircular\_sorting

{

class Program

{

static Node start;

public class Node

{

public int data;

public Node next;

public Node prev;

};

static void insertEnd(int value)

{

Node new\_node;

if (start == null)

{

new\_node = new Node();

new\_node.data = value;

new\_node.next = new\_node.prev = new\_node;

start = new\_node;

return;

}

Node last = (start).prev;

new\_node = new Node();

new\_node.data = value;

new\_node.next = start;

(start).prev = new\_node;

new\_node.prev = last;

last.next = new\_node;

}

static void insertBegin(int value)

{

Node last = (start).prev;

Node new\_node = new Node();

new\_node.data = value;

new\_node.next = start;

new\_node.prev = last;

last.next = (start).prev = new\_node;

start = new\_node;

}

static void insertAfter(int value1, int value2)

{

Node new\_node = new Node();

new\_node.data = value1;

Node temp = start;

while (temp.data != value2)

temp = temp.next;

Node next = temp.next;

temp.next = new\_node;

new\_node.prev = temp;

new\_node.next = next;

next.prev = new\_node;

}

static void display()

{

Node temp = start;

Console.Write("\nTraversal in forward direction \n");

while (temp.next != start)

{

Console.Write("{0} ", temp.data);

temp = temp.next;

}

Console.Write("{0} ", temp.data);

Console.Write("\nTraversal in reverse direction \n");

Node last = start.prev;

temp = last;

while (temp.prev != last)

{

Console.Write("{0} ", temp.data);

temp = temp.prev;

}

Console.Write("{0} ", temp.data);

}

static Node bubbleSort(Node start)

{

int swapped;

Node ptr1;

Node lptr = null;

if (start == null)

return null;

do

{

swapped = 0;

ptr1 = start;

while (ptr1.next != lptr)

{

if (ptr1.data > ptr1.next.data)

{

int t = ptr1.data;

ptr1.data = ptr1.next.data;

ptr1.next.data = t;

swapped = 1;

}

ptr1 = ptr1.next;

}

lptr = ptr1;

}

while (swapped != 0);

return start;

}

static void Main(string[] args)

{

Node start = null;

insertEnd(5);

insertBegin(44);

insertEnd(7);

insertEnd(8);

insertAfter(6, 5);

Console.Write("Created circular doubly linked list is: ");

display();

Console.Write("Sorted List Is ");

bubbleSort(start);

display();

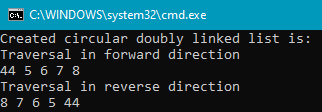
Console.WriteLine();

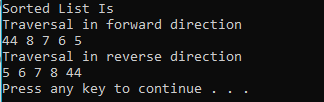
}

}

}

**OUTPUT**:



****

**Task No. 2: Implement dynamic queue using circular linked list.**

**Solution:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace circularlinkedlist

{

}

class DynamicQueue

{

class Node

{

public Node prev;

public int info;

public Node next;

public Node(int i)

{

info = i;

next = null;

prev = null;

}

}

private Node start;

public DynamicQueue()

{

start = null;

}

public void DisplayQueue()

{

Node p;

if (start == null)

{

Console.WriteLine("Queue is Empty");

return;

}

p = start;

Console.Write("Queue is : ");

while (p != null)

{

Console.Write(p.info + " ");

p = p.next;

}

Console.WriteLine();

}

public void InsertInEmptyQueue(int data)

{

Node temp = new Node(data);

start = temp;

}

public void InsertElement(int data)

{

Node p;

Node temp = new Node(data);

p = start;

while (p.next != null)

p = p.next;

p.next = temp;

temp.prev = p;

}

public void CreateQueue()

{

int i, n, data;

Console.Write("Enter the number of elements in Queue : ");

n = Convert.ToInt32(Console.ReadLine());

if (n == 0)

return;

Console.Write("Enter the first element to be inserted in Queue : ");

data = Convert.ToInt32(Console.ReadLine());

InsertInEmptyQueue(data);

for (i = 2; i <= n; i++)

{

Console.Write("Enter next element to be inserted in Queue : ");

data = Convert.ToInt32(Console.ReadLine());

InsertElement(data);

}

}

public void DeleteElement()

{

if (start == null) // Queue is empty

return;

if (start.next == null) // Queue has only one element

{

start = null;

return;

}

start = start.next;

start.prev = null;

}

class Program

{

static void Main(string[] args)

{

int choice, data;

DynamicQueue Queue = new DynamicQueue();

Queue.CreateQueue();

while (true)

{

Console.WriteLine("1.Display Queue");

Console.WriteLine("2.Insert in empty Queue ");

Console.WriteLine("3.Insert an element in the Queue");

Console.WriteLine("4.Delete an element from the Queue");

Console.WriteLine("5.Quit");

Console.Write("Enter your choice : ");

choice = Convert.ToInt32(Console.ReadLine());

if (choice == 5)

break;

switch (choice)

{

case 1:

Queue.DisplayQueue();

break;

case 2:

Console.Write("Enter the element to be inserted : ");

data = Convert.ToInt32(Console.ReadLine());

Queue.InsertInEmptyQueue(data);

break;

case 3:

Console.Write("Enter the element to be inserted : ");

data = Convert.ToInt32(Console.ReadLine());

Queue.InsertElement(data);

break;

case 4:

Queue.DeleteElement();

break;

default:

Console.WriteLine("Wrong choice");

break;

}

Console.WriteLine();

}

Console.WriteLine("Exiting");

}

}

}

**OUTPUT**:

