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 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| SNO | DATE | LAB NO | LAB OBJECTIVE | SIGN |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Bahria University,

Karachi Campus



LAB EXPERIMENT NO.

\_\_\_09\_\_\_\_

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| 1 | **Write a program to create a doubly linked list. Display the linked list.** |
| 2 | **Write a program to create a doubly linked list. Implement any sorting algorithm on the list.** |
|  |  |
|  |  |
|  |  |

Submitted On:

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

(Date: 20/12/20)

**Task No. 1: Write a program to create a doubly linked list. Display the linked list.**

**Solution:**

class Program

{

internal class Dnode

{

internal int data;

internal Dnode prev;

internal Dnode next;

public Dnode(int d)

{

data = d;

prev = null;

next = null;

}

public Dnode()

{

data = 0;

prev = null;

next = null;

}

}

internal class Doublelinkedlist

{

internal Dnode head;

internal void insertatfront(Doublelinkedlist doublelinked, int data)

{

Dnode newnode = new Dnode(data);

newnode.next = doublelinked.head;

newnode.prev = null;

if (doublelinked.head != null)

{

doublelinked.head.prev = newnode;

}

doublelinked.head = newnode;

}

internal void insertatend(Doublelinkedlist doublelinked, int data)

{

Dnode newnode = new Dnode(data);

if (doublelinked.head == null)

{

newnode.prev = null;

doublelinked.head = newnode;

return;

}

Dnode lastnode = getlastnode(doublelinked);

lastnode.next = newnode;

newnode.prev = lastnode;

}

internal Dnode getlastnode(Doublelinkedlist doublelinked)

{

Dnode temp = doublelinked.head;

while (temp.next != null)

{

temp = temp.next;

}

return temp;

}

public void insertAfter(Dnode prev\_node, int new\_data)

{

if (prev\_node == null)

{

Console.WriteLine("The given previous" +

" node cannot be null");

return;

}

Dnode new\_node = new Dnode(new\_data);

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

prev\_node.next = new\_node;

}

public void display()

{

Dnode temp = head;

if (head==null)

{

Console.WriteLine("Linked List Is Empty");

}

else

{

Console.WriteLine("Nodes Of Doubly Linked List");

while (temp!=null)

{

Console.WriteLine(temp.data);

temp=temp.next;

}

}

}

internal void DeleteNodebyKey(Doublelinkedlist doubleLinkedList, int key)

{

Dnode temp = doubleLinkedList.head;

if (temp != null && temp.data == key)

{

doubleLinkedList.head = temp.next;

doubleLinkedList.head.prev = null;

return;

}

while (temp != null && temp.data != key)

{

temp = temp.next;

}

if (temp == null)

{

return;

}

if (temp.next != null)

{

temp.next.prev = temp.prev;

}

if (temp.prev != null)

{

temp.prev.next = temp.next;

}

}

}

static void Main(string[] args)

{

Doublelinkedlist list = new Doublelinkedlist();

Console.WriteLine("======Insert at front======");

list.insertatfront(list, 2);

list.insertatfront(list, 3);

list.insertatfront(list, 5);

list.display();

Console.WriteLine("");

Console.WriteLine("======Insert at end======");

list.insertatend(list, 10);

list.insertatend(list, 18);

list.display();

Console.WriteLine("");

Console.WriteLine("====DELETE=====");

list.DeleteNodebyKey(list, 3);

list.display();

Console.WriteLine("====Insert After Or Mid=====");

list.insertAfter(list.head, 6);

list.insertAfter(list.head.next, 46);

list.insertAfter(list.head.next.next, 106);

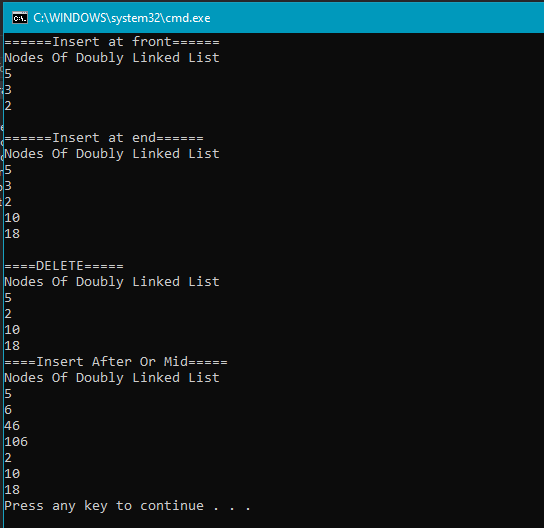
list.display();

}

}

}

**OUTPUT**:



**Task No. 2: Write a program to create a doubly linked list. Implement any sorting algorithm on the list.**

**Solution:**

internal class DoubleLinked

{

// structure of a node

public class Node

{

public int data;

public Node prev;

public Node next;

};

static Node insertAtfront(Node start\_ref, int data)

{

Node ptr1 = new Node();

ptr1.data = data;

ptr1.next = start\_ref;

if (start\_ref != null)

(start\_ref).prev = ptr1;

start\_ref = ptr1;

return start\_ref;

}

static void display(Node start)

{

Node temp = start;

Console.WriteLine();

while (temp != null)

{

Console.Write(temp.data + " ");

temp = temp.next;

}

}

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)

{

int[] arr = { 12, 56, 2, 11, 1, 90 };

int i;

Node start = null;

for (i = 0; i < 6; i++)

{

start = insertAtfront(start, arr[i]);

}

Console.WriteLine("Linked list before sorting");

display(start);

Console.WriteLine();

Console.WriteLine("Linked list after sorting");

start = bubbleSort(start);

display(start);

Console.WriteLine();

**OUTPUT:**

