Task 1: Create an array of length 10 of integers. Values ranging from 1 to 50.

1. Find all pair of elements whose sum is 25.

2. Find the number of elements of A which are even, and the number of elements of A which are odd.

3. Write a procedure which finds the average of the value of A.

Solution:

Random rnd = new Random();

int[] arr = new int[10];

int even=0, odd=0, total=0;

double average;

for (int i = 0; i < arr.Length; i++)

{

arr[i] = rnd.Next(0, 50);

if (arr[i] % 2 == 0)

{

even++;

total+=arr[i];

}

else

{

odd++;

total += arr[i];

}

Console.Write(arr[i]+" ,");

}

average=total/10;

Console.WriteLine();

for (int i = 0; i < arr.Length; i++)

{

for (int y = arr.Length-1; y > i; y--)

{

if (arr[i] + arr[y] == 25)

{

Console.WriteLine(arr[i] + " + " + arr[y] + " = 25");

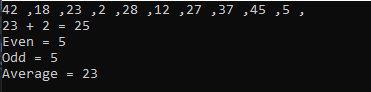
}

}

}

Console.WriteLine("Even = "+even+"\nOdd = "+odd+"\nAverage = "+average);

Output:



Task 2: Write a program which input 2 matrix of user defined rows and columns and perform following operation

 a. Display/Print as a Matrix

b. Addition of Matrix

c. Subtraction of Matrix

d. matrix multiplication

e. Determinant

f. Inverse

Solution:

Random rnd = new Random();

int[,] arr1 = new int[2, 2];

int[,] arr2 = new int[2, 2];

int[,] arr3 = new int[2, 2];

int temp = 0;

for (int x=0; x<2; x++)

{

for(int y=0; y<2; y++)

{

arr1[x,y] = rnd.Next(0,10);

arr2[x, y] = rnd.Next(0,10);

}

}

Console.WriteLine("Matrix 1:");

for (int x = 0; x < 2; x++)

{

for (int y = 0; y < 2; y++)

{

Console.Write(arr1[x, y]+"\t");

}

Console.WriteLine();

}

Console.WriteLine("Matrix 2:");

for (int x = 0; x < 2; x++)

{

for (int y = 0; y < 2; y++)

{

Console.Write(arr2[x, y]+"\t");

}

Console.WriteLine();

}

Console.WriteLine("Addition :");

for (int x = 0; x < 2; x++)

{

for (int y = 0; y < 2; y++)

{

Console.Write(arr2[x, y]+arr1[x,y] + "\t");

}

Console.WriteLine();

}

Console.WriteLine("Subtraction :");

for (int x = 0; x < 2; x++)

{

for (int y = 0; y < 2; y++)

{

Console.Write(arr1[x, y] - arr2[x, y] + "\t");

}

Console.WriteLine();

}

Console.WriteLine("Multiplacation :");

if (arr1.GetLength(1) == arr2.GetLength(0))

{

for (int y = 0; y < arr1.GetLength(0); y++)

{

for (int z = 0; z < arr2.GetLength(0); z++)

{

temp = 0;

for (int x = 0; x < arr1.GetLength(1); x++)

{

temp += arr1[y, x] \* arr2[x, z];

}

arr3[y, z] = temp;

}

}

for (int x = 0; x < 2; x++)

{

for (int y = 0; y < 2; y++)

{

Console.Write(arr3[x, y] + "\t");

}

Console.WriteLine();

}

}

else

{

Console.WriteLine("Error");

}

Console.Write("\nDeterminent of metrix 1:");

int det = arr1[0, 0] \* arr1[1, 1] - arr1[1, 0] \* arr1[0, 1];

Console.WriteLine(det);

Console.WriteLine("Adjoint of matrix 1:");

temp = arr1[0, 0];

arr1[0, 0] = arr1[1, 1];

arr1[1, 1] = temp;

arr1[0, 1] \*= -1;

arr1[1,0] \*= -1;

for (int x = 0; x < 2; x++)

{

for (int y = 0; y < 2; y++)

{

Console.Write(arr1[x, y] + "\t");

}

Console.WriteLine();

}

Console.WriteLine("Inverse of Matrix 1:");

if(det != 0)

{

for (int x = 0; x < 2; x++)

{

for (int y = 0; y < 2; y++)

{

Console.Write(arr1[x, y]+ "/" + det + "\t");

}

Console.WriteLine();

}

}

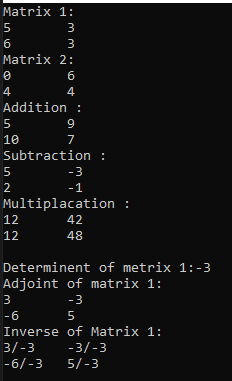
else

{

Console.WriteLine("Error");

}

Output:



Task 1: Which type of sorting you want to apply? Create a menu having the following options:

1. Bubble Sort Method
2. Selection Sort Method
3. Insertion Sort Method

Implement using methods.

Solution:

static void Main(string[] args)

{

Console.Write("Enter array size: ");

int size = int.Parse(Console.ReadLine());

int[] arr = new int[size];

Console.WriteLine("\nENTER ELEMENTS ");

for (int i = 0; i < size; i++)

{

Console.Write("Element {0}: ",i+1);

arr[i] = int.Parse(Console.ReadLine());

}

Console.WriteLine("\nMENU - SELECT A NUMBER");

Console.WriteLine("1. BubbleSort \n2. SelectionSort \n3. InsertionSort");

int menu = int.Parse(Console.ReadLine());

if (menu == 1)

BubbleSort(arr);

else if (menu == 2)

SelectionSort(arr);

else if (menu == 3)

InsertionSort(arr);

else

Console.WriteLine("MENU - INVALID SELECTION");

}

static void BubbleSort(int[] arr)

{

int n = arr.Length;

int k;

for (int m = n; m >= 0; m--)

{

for (int i = 0; i < n - 1; i++)

{

k = i + 1;

if (arr[i] > arr[k])

{

int temp;

temp = arr[i];

arr[i] = arr[k];

arr[k] = temp;

}

}

}

for (int i = 0; i < arr.Length; i++)

Console.Write(" " + arr[i]);

}

static void SelectionSort(int[] arr)

{

int n = arr.Length;

int temp, min;

for (int i = 0; i < n - 1; i++)

{

min = i;

for (int j = i + 1; j < n; j++)

{

if (arr[j] < arr[min])

{

min = j;

}

}

temp = arr[min];

arr[min] = arr[i];

arr[i] = temp;

}

for (int i = 0; i < arr.Length; i++)

Console.Write(" " + arr[i]);

}

static void InsertionSort(int[] arr)

{

for (int i = 0; i < arr.Length - 1; i++)

{

for (int j = i + 1; j > 0; j--)

{

if (arr[j - 1] > arr[j])

{

int temp = arr[j - 1];

arr[j - 1] = arr[j];

arr[j] = temp;

}

}

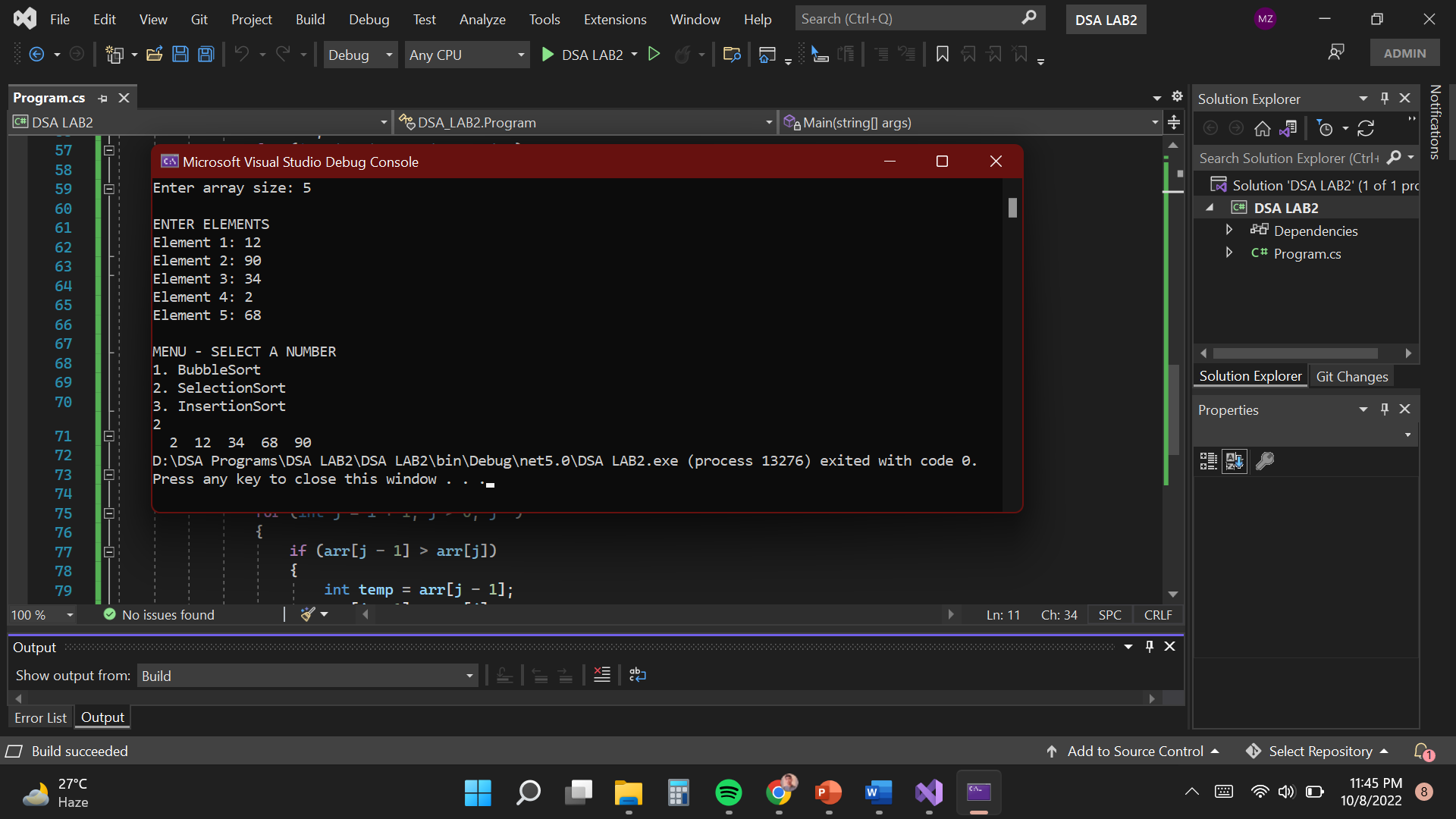
}

for (int i = 0; i < arr.Length; i++)

Console.Write(" " + arr[i]);

}

Output:



Task 2: Implement Selection sort and print string array data in descending order.

Solution:

Console.Write("Enter array size: ");

int size = int.Parse(Console.ReadLine());

string[] arr = new string[size];

Console.WriteLine("\nENTER ELEMENTS ");

for (int i = 0; i < size; i++)

{

Console.Write("Element {0}: ", i + 1);

arr[i] = Console.ReadLine();

}

int n = arr.Length;

string temp;

int max;

for (int i = 0; i < n - 1; i++)

{

max = i;

for (int j = i + 1; j < n; j++)

{

if (int.Parse(arr[j]) > int.Parse(arr[max]))

max = j;

}

temp = arr[max];

arr[max] = arr[i];

arr[i] = temp;

}

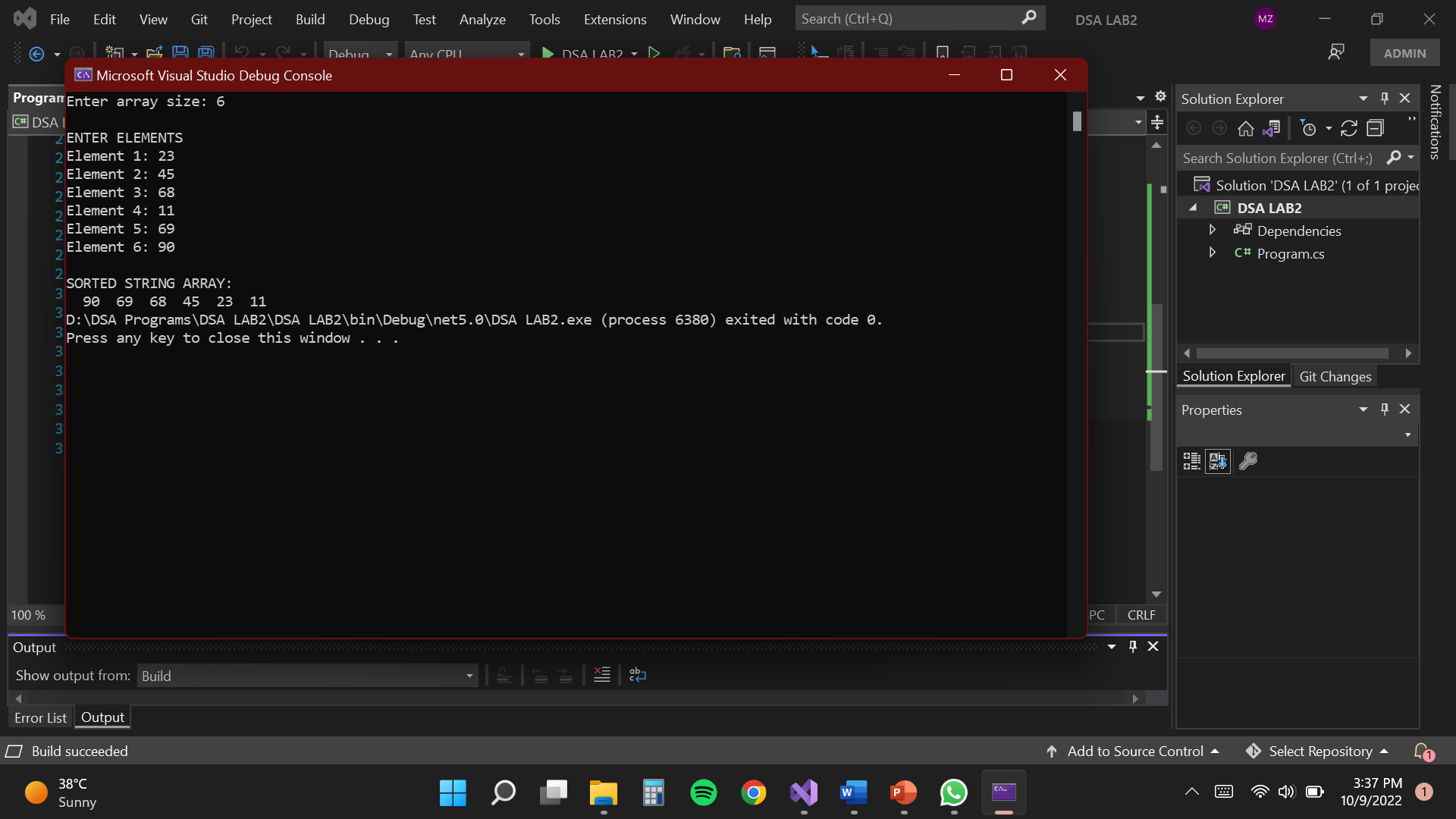
Console.WriteLine("\nSORTED STRING ARRAY:");

for (int i = 0; i < arr.Length; i++)

Console.Write(" " + arr[i]);

}

Output:



Task 3: A Detox chemical Industry has a list of chemicals along with their concentration and Volume. Your task is to list down the name of chemicals in descending order based on their Volume.

Solution:

Console.Write("Enter no. of chemicals: ");

int n = int.Parse(Console.ReadLine());

string[,] arr = new string[n, 3];

for (int i = 0; i < n; i++)

{

Console.Write("\nEnter chemical name: ");

arr[i, 0] = Console.ReadLine().ToUpper();

Console.Write("Enter concentration of {0}: ",arr[i,0]);

arr[i, 1] = Console.ReadLine();

Console.Write("Enter volume of {0}: ",arr[i,0]);

arr[i, 2] = Console.ReadLine();

}

Console.WriteLine("\nUNSORTED CHEMICAL LIST: ");

Console.WriteLine("\nNAME\t\tCONCENTRATION\tVOLUME");

for (int i = 0; i < n; i++)

Console.WriteLine(arr[i, 0] + "\t" + arr[i, 1] + "\t" + arr[i,2]);

Console.WriteLine("\nSORTED CHEMICAL LIST (wrt to volume): ");

int k;

for (int m = n; m >= 0; m--)

{

for (int i = 0; i < n - 1; i++)

{

k = i + 1;

if (Convert.ToDouble(arr[i, 2]) < Convert.ToDouble(arr[k, 2]))

{

string name, conc, vol;

name = arr[i, 0];

conc = arr[i, 1];

vol = arr[i, 2];

arr[i, 0] = arr[k, 0];

arr[i, 1] = arr[k, 1];

arr[i, 2] = arr[k, 2];

arr[k, 0] = name;

arr[k, 1] = conc;

arr[k, 2] = vol;

}

}

}

Console.WriteLine("\nNAME\t\tCONCENTRATION\tVOLUME");

for (int i = 0; i < n; i++)

Console.WriteLine(arr[i, 0] + "\t" + arr[i, 1] + "\t" + arr[i, 2]);

Console.WriteLine("\nSEARCH CHEMICALS");

Console.Write("Enter chemical name: ");

string search = Console.ReadLine().ToUpper();

for (int i = 0; i < n; i++)

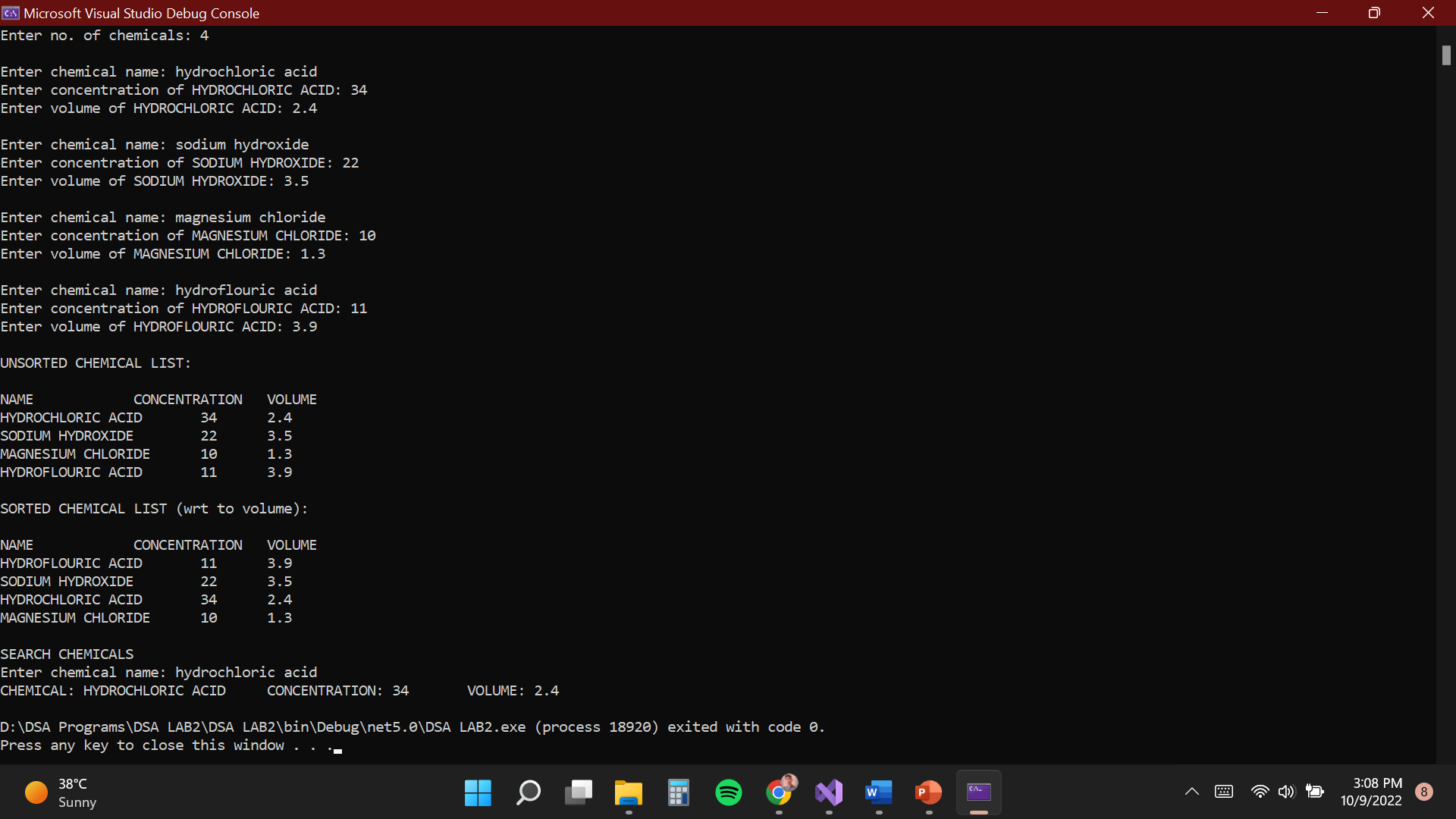
{

if (search == arr[i, 0])

Console.WriteLine("CHEMICAL: "+arr[i, 0] +"\tCONCENTRATION: "+ arr[i, 1]+"\tVOLUME: "+arr[i, 2]);

}

Output:



Task 4: You have to write a program which take input from the user and place the value on correct location in ascending order.

Solution:

Console.Write("Enter array size: ");

int size = int.Parse(Console.ReadLine());

int[] arr = new int[size];

int n = arr.Length;

Console.WriteLine("ENTER ELEMENTS ");

for (int i = 0; i < n; i++)

{

Console.Write("Element {0}: ",i+1);

arr[i] = int.Parse(Console.ReadLine());

}

Console.WriteLine("\nUNSORTED ARRAY:");

for(int i = 0;i<n;i++)

Console.WriteLine("INDEX {0}: {1}",i,arr[i]);

Console.WriteLine("\nSORTED ARRAY WITHOUT INSERTING NEW ELEMENT: ");

SelectionSort(arr);

int[] arr2 = new int[n+1];

Console.Write("\nInsert a new element: ");

int insert = int.Parse(Console.ReadLine());

for(int i=0; i < arr2.Length; i++)

{

if (i<n)

arr2[i] = arr[i];

else

arr2[i] = insert;

}

Console.WriteLine("\nSORTED ARRAY WITH INSERTING NEW ELEMENT: ");

SelectionSort(arr2);

}

static void SelectionSort(int[] arr)

{

int n = arr.Length;

int temp, min;

for (int i = 0; i < n - 1; i++)

{

min = i;

for (int j = i + 1; j < n; j++)

{

if (arr[j] < arr[min])

{

min = j;

}

}

temp = arr[min];

arr[min] = arr[i];

arr[i] = temp;

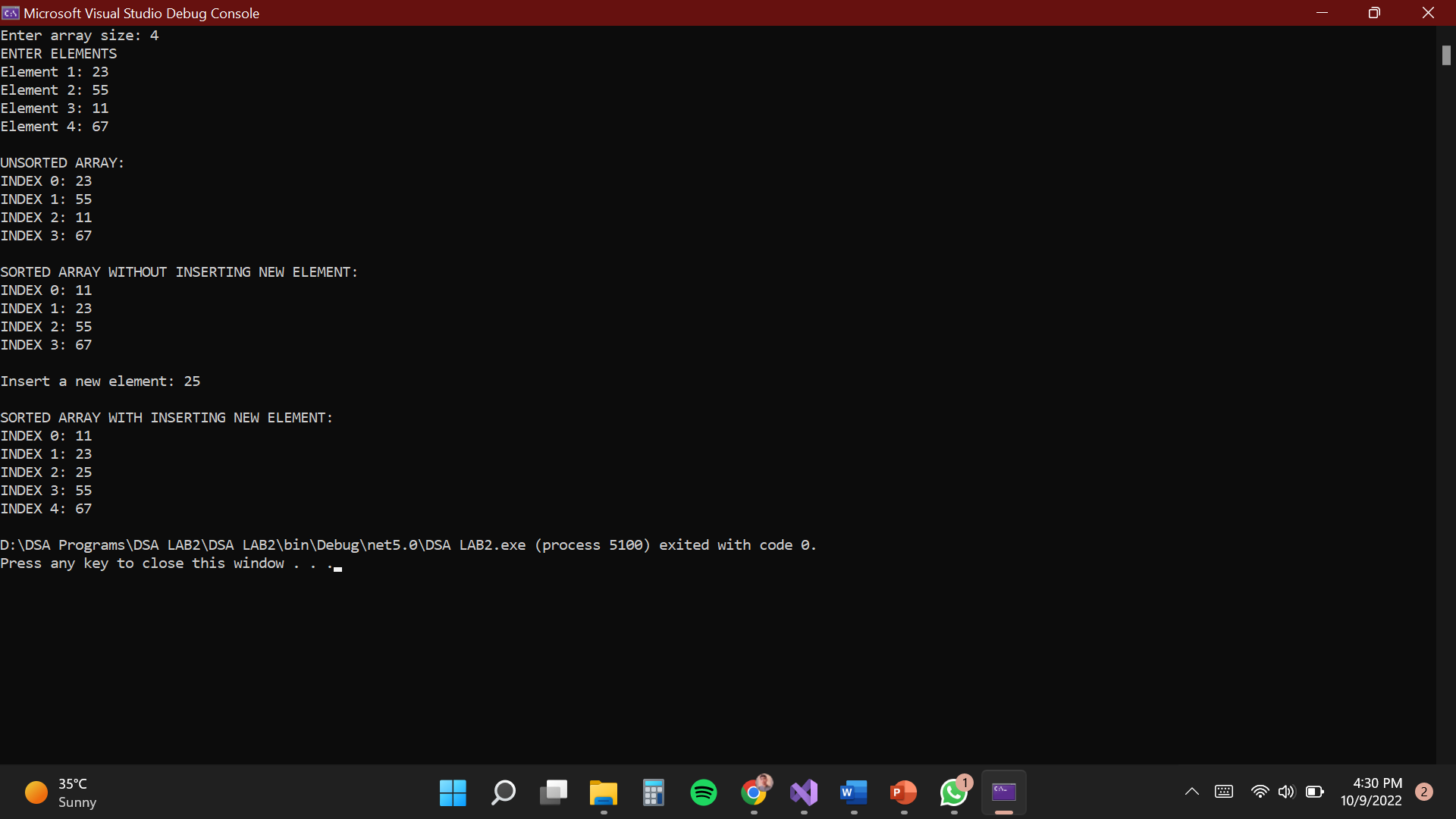
}

for (int i = 0; i < n; i++)

Console.WriteLine("INDEX {0}: {1}", i, arr[i]);

}

Output:



Task 5: Write a program which take N numbers of grocery items from user along with their price. Your main task is to display the items in sorted format. Then allow user to search for any of the item from that list by using name of the item.

Solution:

Console.Write("Enter no. of grocery items: ");

int n = int.Parse(Console.ReadLine());

string[,] arr = new string[n, 2];

for (int i = 0; i < n; i++)

{

Console.Write("\nEnter name of item: ");

arr[i, 0] = Console.ReadLine().ToUpper();

Console.Write("Enter price of {0}: ",arr[i,0]);

arr[i, 1] = Console.ReadLine();

}

Console.WriteLine("UNSORTED ARRAY: ");

for (int i = 0; i < n; i++)

Console.WriteLine(arr[i, 0] + "\t$" + arr[i, 1]);

Console.WriteLine("\nSORTED ARRAY:");

int k;

for (int m = n; m >= 0; m--)

{

for (int i = 0; i < n - 1; i++)

{

k = i + 1;

if (int.Parse(arr[i, 1]) > int.Parse(arr[k, 1]))

{

string name, price;

name = arr[i, 0];

price = arr[i, 1];

arr[i, 0] = arr[k, 0];

arr[i, 1] = arr[k, 1];

arr[k, 0] = name;

arr[k, 1] = price;

}

}

}

for (int i = 0; i < n; i++)

Console.WriteLine(arr[i, 0] + "\t$" + arr[i, 1]);

Console.WriteLine("\nSEARCH ITEMS");

Console.Write("Enter name of the item: ");

string search = Console.ReadLine().ToUpper();

for (int i = 0; i < n; i++)

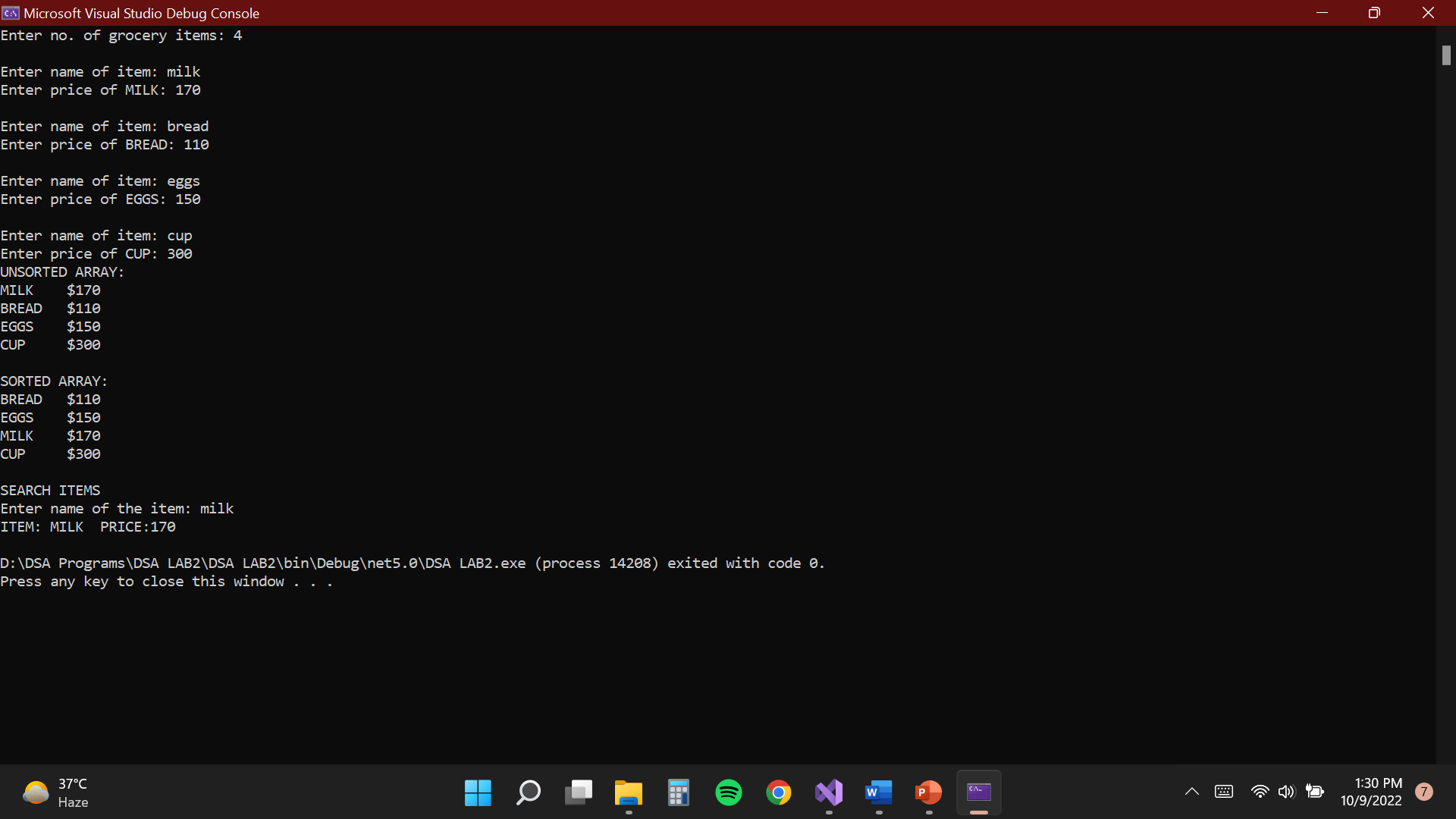
{

if (search == arr[i, 0])

Console.WriteLine("ITEM: "+arr[i, 0] +" PRICE:"+ arr[i, 1]);

}

Output:



Task 1: Write a program to create a linked list and pertform

\*traversing

\* Insertion

\*deletion

Solution:

Main:

using Task\_4;

Console.WriteLine("Task 4");

DLL dll = new DLL();

dll.head = new Node(5);

Node second = new Node(6);

Node thired = new Node(7);

Node fourth = new Node(8);

dll.head.next = second;

second.next = thired;

thired.next = fourth;

fourth.pre = thired;

thired.pre = second;

second.pre = dll.head;

Console.WriteLine("\n\nDouble Linked List :");

dll.display(dll.head);

Console.WriteLine("\n\nPush 9 :");

dll.push(9);

dll.display(dll.head);

Console.WriteLine("\n\nInsert after 2 6 :");

dll.insertAfter(second, 2);

dll.display(dll.head);

Console.WriteLine("\n\nAppend 99 :");

dll.append(99);

dll.display(dll.head);

Console.WriteLine("\n\nDelete head XD :");

dll.delete(dll.head);

dll.display(dll.head);

Node:

namespace Task\_4

{

public class Node

{

public int data;

public Node next;

public Node pre;

public Node(int data)

{

this.data = data;

}

}

}

Double Linked List:

namespace Task\_4

{

public class DLL

{

public Node head;

public void display(Node key)

{

Node last = null;

if(key == null)

{

Console.WriteLine("Invalid Node");

}

else

{

while(key != null)

{

Console.Write(key.data + " ,");

last = key;

key = key.next;

}

while(last != null)

{

Console.Write(last.data+" ,");

last = last.pre;

}

}

}

public void push(int data)

{

Node nNode = new Node(data);

if (head != null)

{

nNode.next = head;

nNode.next.pre = nNode;

head = nNode;

nNode.pre = null;

}

else

{

head = nNode;

}

}

public void insertAfter(Node pNode, int data)

{

Node nNode = new Node(data);

if(pNode != null)

{

pNode.next.pre = nNode;

nNode.next = pNode.next;

nNode.pre = pNode;

pNode.next = nNode;

}

else

{

Console.WriteLine("Invalid Node");

}

}

public void append(int data)

{

Node nNode = new Node(data);

Node temp = head;

if(head != null) {

while (temp.next != null)

{

temp = temp.next;

}

temp.next = nNode;

nNode.pre =temp;

nNode.next = null;

}

else

{

head = nNode;

}

}

public void delete(Node key)

{

if (key == null)

{

Console.WriteLine("Invalid Node");

}

else

{

if(key == head)

{

head = head.next;

head.pre = null;

}

else

{

key.pre.next = key.next;

key.next.pre = key.pre;

}

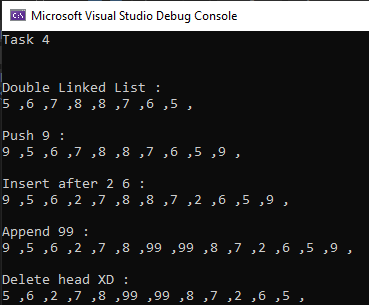
}

}

}

}

Output:



Task 1 Write a program to build your own stack class. The minimum your stack class should include is using your age .

* A **Push(Object)** method
* A **Pop()** method
* A **Peek()** method
* A **IsFull()** method
* A **IsEmpty()** method
* A **Display()** method
* A **Count()** method

Solution:

Main:

using Task\_5;

stack myStack = new stack();

myStack.push(19);

myStack.push(22);

myStack.push(21);

myStack.push(20);

Console.WriteLine("4 ITEMS flUSHED TO THE STACK: ");

myStack.Display();

myStack.Count();

myStack.peek();

myStack.pop();

Console.WriteLine("\nSTACK AFTER flOflflING AN ITEM: ");

myStack.Display();

myStack.Count();

Node:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Task\_4

{

internal class stack

{

static readonly int MAX = 20;

int top, count = 0;

int[] stk = new int[MAX];

bool IsEmpty()

{

return (top < 0);

}

bool IsFull()

{

return (top > MAX);

}

public stack()

{

top = -1;

}

internal bool push(int data)

{

stk[++top] = data;

count++;

return true;

}

internal int pop()

{

if(top < 0)

{

Console.WriteLine("Stack Underflow");

return 0;

}

else

{

int value = stk[top--];

count--;

return 0;

}

}

internal void peek()

{

if(top < 0)

{

Console.WriteLine("Stack Underflow");

return;

}

else

Console.WriteLine("The topmost element of stack is: {0}", stk[top]);

}

internal void Display()

{

if(top < 0)

{

Console.WriteLine("Stack Underflow"); return;

}

else

{

for (int i = top; i >= 0; i--)

{

Console.WriteLine(stk[i]);

}

}

}

internal void Count()

{

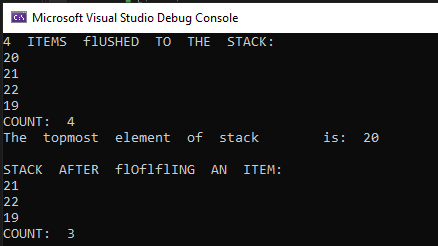
Console.WriteLine("COUNT: {0}", count);

}

}

}

Output:



Task 2: Write a program that implements stack using linked list.

Solution:

Main:

LinkedList stack = new LinkedList();

stack.head = new Node(20);

Node second = new Node(21);

Node third = new Node(22);

Node fourth = new Node(19);

stack.head.next = second;

second.next = third;

third.next = fourth;

Console.WriteLine("ELEMENTS IN THE STACK");

stack.display();

stack.push(23);

Console.WriteLine("\nELEMENTS IN THE STACK AFTER PUSHING AN ELEMENT");

stack.display();

stack.pop();

stack.pop();

stack.pop();

Console.WriteLine("\nELEMENTS IN THE STACK AFTER POPPING THREE ELEMENTS");

stack.display();

Node:

public int data;

public Node next;

public Node(int d)

{

data = d;

next = null;

}

LinkedList:

public int data;

public Node next;

public Node(int d)

{

data = d;

next = null;

}

}

public class LinkedList

{

public Node head;

public void display()

{

Node n = head;

while (n != null)

{

Console.WriteLine(n.data + " ");

n = n.next;

}

}

public void push(int new\_data)

{

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

if (head == null)

{

head = new Node(new\_data);

return;

}

new\_node.next = null;

Node last = head;

while (last.next != null)

last = last.next;

last.next = new\_node;

return;

}

public void pop()

{

Node temp = null;

Node n = head;

while (n.next != null)

{

temp = n;

n = n.next;

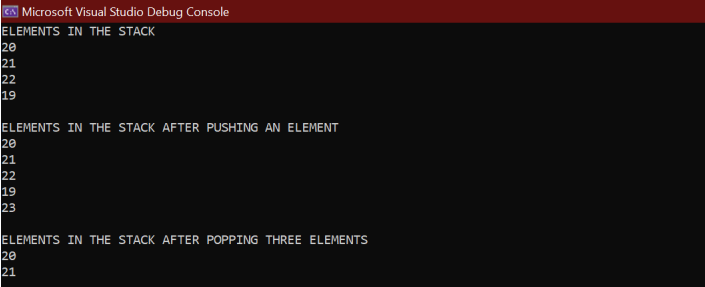
}

temp.next = null;

}

}

Output:



Task 1: Create a Messaging System using Queue which makes use of a buffer to store messages to be sent if they are more then 5.

Solution:

using System.Collections;

namespace Task\_7

{

class Program

{

static void Main(string[] args)

{

Queue msgs = new Queue();

Queue buffer = new Queue();

int menu;

do

{

Console.WriteLine("Select Option");

Console.WriteLine("1.ENQUEUE \n2.DEQUEUE \n3.DISPLAY \n4.EXIT");

try

{

menu = int.Parse(Console.ReadLine());

}

catch(Exception ex)

{

Console.WriteLine("Wrong Option, Select again");

Console.WriteLine("1.ENQUEUE \n2.DEQUEUE \n3.DISPLAY \n4.EXIT");

menu = int.Parse(Console.ReadLine());

}

if (menu == 1)

{

Console.WriteLine("ENQUEUE");

Console.Write("Enter msg to send: ");

string m = Console.ReadLine();

if (msgs.isFull(msgs.arr, msgs.rear))

{

Console.WriteLine("QUEUE IS FULL - STORING IN BUFFER");

buffer.enqueue(m);

}

else

msgs.enqueue(m);

}

else if (menu == 2)

{

Console.WriteLine("DEQUEUE");

msgs.dequeue();

if (!buffer.isEmpty(buffer.front, buffer.rear))

{

msgs.enqueue(buffer.arr[0]);

buffer.dequeue();

}

}

else if (menu == 3)

{

Console.WriteLine("DISPLAYING FROM QUEUE");

msgs.display();

}

else if (menu == 4)

{

Console.WriteLine("EXITING PROGRAM..");

break;

}

else

{

Console.WriteLine("Invalid option, Try again");

}

}

while (true);

}

}

}

Queue:

using System;

namespace dsalab7

{

class Queue

{

public string[] arr = new string[5];

public int front = -1, rear = -1;

public bool isEmpty(int front, int rear)

{

if (rear == -1 && front == -1)

return true;

else

return false;

}

public bool isFull(string[] arr, int rear)

{

if (rear == arr.Length - 1)

return true;

else

return false;

}

public void enqueue(string x)

{

if (isFull(arr, rear) == true)

{

Console.WriteLine("QUEUE IS FULL - ADDING TO BUFFER");

}

else if (isEmpty(front,rear) == true)

{

front = 0;

rear = 0;

arr[rear] = x;

}

else

{

rear += 1;

arr[rear] = x;

}

}

public void dequeue()

{

if (isEmpty(front, rear) == true)

{

Console.WriteLine("QUEUE IS EMPTY");

return;

}

else if (front == rear)

front = rear = -1;

else

{

for(int i=0; i < 4; i++)

arr[i] = arr[i + 1];

rear = rear - 1;

}

}

public void display()

{

for (int i = 0; i <= rear; i++)

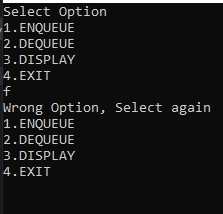
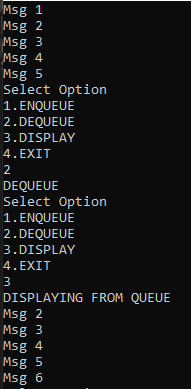
Console.WriteLine(arr[i]);

}

}

}

Output:



Task 1: Quick Sort

Solution:

public static void Main(String[] args)

{

string[] arr = { "bcda", "bwjc", "knlas", "aa", "abc", "hjiw", "dec", "z" };

printarr(arr);

quickSort(arr, 0, 7);

printarr(arr);

}

public static void printarr(string[] arr)

{

for(int i=0; i<arr.Length; i++)

{

Console.Write(arr[i] + " ,");

}

Console.WriteLine();

}

public static int Partition(string[] arr, int low, int high)

{

string pivot = arr[high];

int i = low-1;

for(int j = low; j<= high-1; j++)

{

if (StringLE(arr[j], pivot))

{

i++;

string temp;

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

string temp1;

temp1 = arr[i+1];

arr[i+1] = arr[high];

arr[high] = temp1;

return i + 1;

}

public static void quickSort(string[] arr,int low, int high)

{

if(low < high)

{

int pi = Partition(arr, low, high);

quickSort(arr, low, pi - 1);

quickSort(arr, pi+1, high);

}

}

public static bool StringLE(string s1, string s2)

{

s1 = s1.ToLower();

s2 = s2.ToLower();

char[] c1 = s1.ToCharArray();

char[] c2 = s2.ToCharArray();

if (s1.Equals(s2))

{

return true;

}

else

{

int length1 = c1.Length;

int length2 = c2.Length;

if(length1 < length2)

{

return true;

}

else if(length1 > length2) {

return false;

}

else

{

for (int i = 0; i < length1; i++)

{

if (c1[i] == c2[i])

{

continue;

}

else if ((int)c1[i] < (int)c2[i])

{

return true;

}

else

{

return false;

}

}

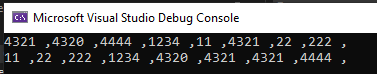
}

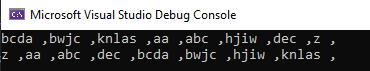
return true;

}

}

Output:





Task 2: Merge Sort

Solution:

Main:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Task\_8;

namespace Task\_8

{

class Program

{

public static void Main(String[] args)

{

int[] arr = { 12, 11, 13, 5, 6, 7 , 56, 24, 5, 12, 6, 41, 65, 65, 0, -5};

Console.WriteLine("Given Array");

printArray(arr);

MergeSort ob = new MergeSort();

ob.sort(arr, 0, arr.Length - 1);

Console.WriteLine("\nSorted array");

printArray(arr);

}

public static void printArray(int[] arr)

{

int n = arr.Length;

for (int i = 0; i < n; ++i)

Console.Write(arr[i] + " ");

Console.WriteLine();

}

}

}

Merge Sort Class:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Task\_8

{

public class MergeSort

{

public void sort(int[] arr, int l, int r)

{

if (l < r)

{

int m = l + (r - l) / 2;

sort(arr, l, m);

sort(arr, m + 1, r);

merge(arr, l, m, r);

}

}

public void merge(int[] arr, int l, int m, int r)

{

int n1 = m - l + 1;

int n2 = r - m;

int[] L = new int[n1];

int[] R = new int[n2];

int i, j;

// Copy data to temp arrays

for (i = 0; i < n1; ++i)

L[i] = arr[l + i];

for (j = 0; j < n2; ++j)

R[j] = arr[m + 1 + j];

i = 0;

j = 0;

int k = l;

while (i < n1 && j < n2)

{

if (L[i] <= R[j])

{

arr[k] = L[i];

i++;

}

else

{

arr[k] = R[j];

j++;

}

k++;

}

while (i < n1)

{

arr[k] = L[i];

i++;

k++;

}

while (j < n2)

{

arr[k] = R[j];

j++;

k++;

}

}

}

}

Solution:

