 **BAHRIA UNIVERSITY (KARACHI CAMPUS)**

**ASSGINMENT # 1 - FALL 2021**

# Data Structures and Algorithm (CSC-221)

Class: **BSE 3-B** Submission Deadline: **2nd Feb, 2022**

Course Instructor: **Engr. Laraib Siddique** CLO Mapping: **CLO 1**

Lab Instructor: **Engr. Ayesha Khan** Max Marks: **10**

Name: **Muhammad Junaid Saleem Qadri** E.no : **02-131202-057**

1. In a Bottle Coloring Factory, items are moving on a conveyor belt. A mechanical arm will sort them based on the items labels and place the bottles on the required section to be colored. **[6 marks]**

The bottles are colored as follows:

* If the bottle is labelled in the range A-K, then color it RED.
* If the bottle is labelled in the range L-S, then color it BLUE.
* If the bottle is labelled in the range T-Z, then color it GREEN.

Make a random function to generate labels for the bottles.

After coloring the bottles are placed in boxes. Mechanical arm will first take five bottles to be colored then put four colored bottles in boxes based on priority. The priorities for removal of colored bottles are first Green, second Blue, and third Red. The above process should be repeated six times. At the end of the program no bottle left in the factory.

**Solution:-**

**Class Factory**

class Factory

{

int bottles;

char[] bottleLabels;

string[] bottleColors;

int colorstrackerRear;

int colorstrackerFront;

char[] boxesQueue;

string[] boxes;

int boxestracker;

int boxFront;

int boxRear;

int boxno = 1;

public Factory(int bottles)

{

this.bottles = bottles;

bottleLabels = new char[bottles];

bottleColors = new string[bottles];

boxes = new string[bottles / 4]; // a box contain only 4 bottles

boxesQueue = new char[4];

colorstrackerRear = colorstrackerFront = boxestracker = boxFront

= boxRear = 0;

}

public void AssignLabels()

{

Random rnd = new Random();

Console.WriteLine("Bottles in the conveyor belt are:-");

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

{

int asciiCode = rnd.Next(65, 91);

bottleLabels[i] = Convert.ToChar(asciiCode);

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

}

Console.WriteLine("\n");

}

public void AssignColors()

{

if (colorstrackerRear < bottles)

{

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

{

if (colorstrackerRear < bottles)

{

if (bottleLabels[colorstrackerRear] >= 'A' &&

bottleLabels[colorstrackerRear] <= 'K')

bottleColors[colorstrackerRear] = "RED";

else if (bottleLabels[colorstrackerRear] >= 'L' &&

bottleLabels[colorstrackerRear] <= 'S')

bottleColors[colorstrackerRear] = "BLUE";

else

bottleColors[colorstrackerRear] = "GREEN";

colorstrackerRear++;

}

}

}

}

public void AddToBoxQueue()

{

char[] templqueue = new char[4];

string[] tempcqueue = new string[4];

boxRear = 0;

if (colorstrackerFront <= colorstrackerRear)

{

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

{

if (colorstrackerFront < colorstrackerRear)

{

templqueue[i] = bottleLabels[colorstrackerFront];

tempcqueue[i] = bottleColors[colorstrackerFront];

colorstrackerFront++;

}

}

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

{

if (string.Equals(tempcqueue[i], "RED",

StringComparison.OrdinalIgnoreCase) == true)

{

boxesQueue[boxRear] = templqueue[i];

boxRear++;

}

}

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

{

if (string.Equals(tempcqueue[i], "BLUE",

StringComparison.OrdinalIgnoreCase) == true)

{

boxesQueue[boxRear] = templqueue[i];

boxRear++;

}

}

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

{

if (string.Equals(tempcqueue[i], "GREEN",

StringComparison.OrdinalIgnoreCase) == true)

{

boxesQueue[boxRear] = templqueue[i];

boxRear++;

}

}

char box;

Console.Write("Box " + boxno + ": ");

boxno++;

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

{

box = boxesQueue[i];

if(box>='A' && box <= 'K')

{

Console.ForegroundColor = ConsoleColor.Red;

}

else if(box>='L' && box <= 'S')

{

Console.ForegroundColor= ConsoleColor.Blue;

}

else

Console.ForegroundColor= ConsoleColor.Green;

Console.Write(box+" ");

}

Console.ResetColor();

Console.WriteLine();

}

}

public void ConveyorBelt()

{

AssignLabels(); // Batch 1

AssignColors();

AddToBoxQueue(); // Batch 2

AssignColors();

AddToBoxQueue(); // Batch 3

AssignColors();

AddToBoxQueue(); // Batch 4

AssignColors();

AddToBoxQueue(); // Batch 5

AssignColors();

AddToBoxQueue(); // Batch 6

AssignColors();

AddToBoxQueue();

Console.WriteLine("\nNo more bottles left");

}

}

**Output:-**

A screenshot of a computer screen

Description automatically generated with medium confidence

1. Write an algorithm that prints out all the subset of four elements of a set of n elements the elements of this set are sorted in a list that is input to the algorithm. **[2 marks]**

**Solution:-**

static void Subset(int[] arr, int n, int r, int index, int[] temp, int i)

{

if (index == r)

{

for (int j = 0; j < r; j++)

Console.Write(temp[j] + " ");

Console.WriteLine("");

return;

}

if (i >= n) return;

temp[index] = arr[i];

Subset(arr, n, r, index + 1, temp, i + 1);

Subset(arr, n, r, index, temp, i + 1);

}

static void printSubsets(int[] arr, int n, int r)

{

int[] temp = new int[r];

Subset(arr, n, r, 0, temp, 0);

}

static void Display(int[] arr)

{

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

{

Console.Write("Enter Input {0} = ", i + 1);

arr[i] = Convert.ToInt32(Console.ReadLine());

}

Console.Write("Original Set : { ");

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

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

Console.Write("}");

Console.WriteLine("\n\nSub Sets are : ");

}

static void Main(string[] args)

{

Console.Write("Enter number of input = ");

int input = Convert.ToInt32(Console.ReadLine());

int[] list = new int[input];

Display(list);

printSubsets(list, list.Length, 4);

}

A screenshot of a computer

Description automatically generated with medium confidence**Output:-**

1. Take 10 inputs from the user and assign them into two string arrays (make 2 unsorted string arrays of 5 lengths each), merge those arrays and obtain the result in the sorted manner. **[2 marks]**

**Solution:-**

public static string[] mergeSort(string[] array)

{

string[] left;

string[] right;

string[] result = new string[array.Length];

if (array.Length <= 1)

return array;

int midPoint = array.Length / 2;

left = new string[midPoint];

if (array.Length % 2 == 0)

right = new string[midPoint];

else

right = new string[midPoint + 1];

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

left[i] = array[i];

int x = 0;

for (int i = midPoint; i < array.Length; i++)

{

right[x] = array[i];

x++;

}

left = mergeSort(left);

right = mergeSort(right);

result = merge(left, right);

return result;

}

public static string[] merge(string[] left, string[] right)

{

int resultLength = right.Length + left.Length;

string[] result = new string[resultLength];

int indexLeft = 0, indexRight = 0, indexResult = 0;

while (indexLeft < left.Length || indexRight < right.Length)

{

if (indexLeft < left.Length && indexRight < right.Length)

{

if (left[indexLeft].CompareTo(right[indexRight])<=0)

{

result[indexResult] = left[indexLeft];

indexLeft++;

indexResult++;

}

else

{

result[indexResult] = right[indexRight];

indexRight++;

indexResult++;

}

}

else if (indexLeft < left.Length)

{

result[indexResult] = left[indexLeft];

indexLeft++;

indexResult++;

}

else if (indexRight < right.Length)

{

result[indexResult] = right[indexRight];

indexRight++;

indexResult++;

}

}

return result;

}

static void Main(string[] args)

{

string[] array1 = new string[5];

string[] array2 = new string[5];

string[] result = new string[10];

for (int i = 0; i < 10; i++)

{

Console.Write($"Enter Element {i+1} : ");

if (i < 5) array1[i] = Console.ReadLine();

else array2[i - 5] = Console.ReadLine();

}

array1 = mergeSort(array1);

array2 = mergeSort(array2);

result = merge(array1, array2);

Console.WriteLine("\n------ Sorted Array --------");

foreach(string s in result)

{

Console.Write(s+" ");

}

Console.ReadKey();

}

**Output:-**

Graphical user interface, text

Description automatically generated