**Question 01:**

**Code:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DSAassign

{

    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 can hold only 4 bottles

            boxesQueue = new char[4];

            colorstrackerRear = colorstrackerFront = boxestracker = boxFront = boxRear = 0;

        }

        public void AssignLabels()

        {

            Random rnd = new Random();

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

            {

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

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

            }

        }

        public void AssignColors()

        {

            // mech arm takes only five bottles at a time to be colored

            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++;

                        }

                }

                string box = null;

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

                {

                    box += boxesQueue[i] + " ";

                }

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

                boxno++;

                Console.WriteLine(box);

            }

        }

        public void RunConveyorBelt()

        {

            AssignLabels();

            // iteration 1

            AssignColors();

            AddToBoxQueue();

            // iteration 2

            AssignColors();

            AddToBoxQueue();

            // iteration 3

            AssignColors();

            AddToBoxQueue();

            // iteration 4

            AssignColors();

            AddToBoxQueue();

            // iteration 5

            AssignColors();

            AddToBoxQueue();

            // iteration 6

            AssignColors();

            AddToBoxQueue();

            // when everything is done

            Console.WriteLine("No more bottles left");

        }

    }

    class Program

    {

        static void Main(string[] args)

        {

            Factory fac = new Factory(24);

            fac.RunConveyorBelt();

        }

    }

}

**Output:**

**Text

Description automatically generated**

**Question 02:**

**Algorithm:**

1. A list of n integers is given as input to the algorithm.
2. Create a new array temp to hold our subsets.
3. We traverse through the given list of elements.
4. If the element is included in our current combination, we put the element in temp and we recursively call the method with both the indexes of our temp and list incremented.
5. If the element is not included in the current combination, we do not put in the temp array.
6. When the number of elements in temp array becomes equal to ‘4’ we print it.
7. Repeat the steps till we have all the possible combinations of length ‘4’.

**Code:**

using System;

namespace ConsoleApp18

{

class Program

{

static void SubsetsRecursion(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];

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

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

}

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

{

int[] temp = new int[r];

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

}

static void Display(int[] arr)

{

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

{

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

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

}

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

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

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

Console.Write("}");

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

}

static void Main(string[] args)

{

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

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

int[] list = new int[j];

Display(list);

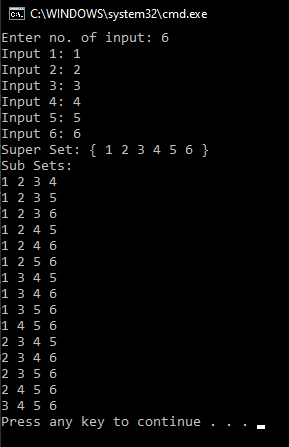
DisplaySubsets(list, list.Length, 4);

}

}

}

**Output:**



**Question 03:**

**Code:**

using System;

namespace ConsoleApp18

{

class Worker

{

public string[] Merge(string[] L, string[] R)

{

int i, j, k;

i = j = k = 0;

int n1 = L.Length;

int n2 = R.Length;

string[] arr = new string[n1 + n2];

while (i < n1 && j < n2)

{

if (string.Equals(L[i],R[j],StringComparison.OrdinalIgnoreCase) == true)

{

arr[k] = L[i];

i++;

}

else

{

arr[k] = R[j];

j++;

}

k++;

}

while (i < n1)

{

arr[k] = L[i];

i++;

k++;

}

while (i < n2)

{

arr[k] = R[j];

j++;

k++;

}

return arr;

}

public string[] BubbleSort(string[] arr)

{

int l = arr.Length;

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

{

for (int j = 0; j < l - 1; j++)

{

if (arr[j].CompareTo(arr[j + 1]) > 0)

{

string temp = arr[j];

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

arr[j + 1] = temp;

}

}

}

return arr;

}

public string[] MergeandSort(string[] L, string[] R)

{

return BubbleSort(Merge(L, R));

}

}

class Program

{

static void UserInput(string[] arr)

{

for (int i = 0; i < 5; i++)

{

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

arr[i] = Console.ReadLine();

}

}

static void PrintArray(string[] arr)

{

Console.WriteLine();

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

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

Console.WriteLine();

}

static void Main(string[] args)

{

string[] userinput1 = new string[5];

string[] userinput2 = new string[5];

Console.WriteLine("Array 1");

UserInput(userinput1);

Console.WriteLine("Array 2");

UserInput(userinput2);

Worker wk = new Worker();

string[] finalarr = wk.MergeandSort(userinput1, userinput2);

PrintArray(finalarr);

}

}

}

**Output:**

