 **BAHRIA UNIVERSITY (KARACHI CAMPUS)**

**ASSGINMENT # 1 - FALL 2021**

# Data Structures and Algorithm (CSC-221)

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

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

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

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.

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]**
2. 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]**

**Name:- Mustufa Class:- BSE III-B Date:- 30/01/2022 Enrollment:- 02-131202-012 Registration:- 69966**

**Question 1:-**

**Solution:-**

class Bottle {

public char label;

public char color;

public Bottle(char l)

{

label = l;

color = ' ';

}

public char getLabel()

{

return label;

}

}

struct element

{

public Bottle bottle;

public int priority;

}

class PriorityQueue

{

element[] elements = new element[5];

int size = -1;

public void enqueue(Bottle b)

{

size++;

elements[size].bottle = b;

if (b.label >= 'a' && b.label <= 'k')

{

elements[size].bottle.color = 'R';

elements[size].priority = 1;

}

else if (b.label >= 'l' && b.label <= 's')

{

elements[size].bottle.color = 'B';

elements[size].priority = 2;

}

else

{

elements[size].bottle.color = 'G';

elements[size].priority = 3;

}

}

public int peek()

{

int highestPriority = 1;

int index = -1;

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

{

if(highestPriority == elements[i].priority && index > -1 && elements[index].bottle.label < elements[i].bottle.label)

{

highestPriority = elements[i].priority;

index = i;

}

else if(highestPriority <= elements[i].priority)

{

highestPriority = elements[i].priority;

index = i;

}

}

return index;

}

public void dequeue()

{

int index = peek();

if(index != -1) {

Console.Write(elements[index].bottle.label+"\t");

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

{

elements[i] = elements[i + 1];

}

}

size--;

}

}

static void Main(string[] args)

{

Bottle[] bottles = new Bottle[30];

Random r = new Random();

PriorityQueue pq = new PriorityQueue();

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

for (int i = 0;i<30;i++)

{

bottles[i] = new Bottle((char)r.Next(97, 123));

Console.Write(bottles[i].getLabel()+"\t");

}

int k = 0;

for(int j = 0;j<6;j++)

{

Console.WriteLine("\n\nBottles colored in {0}th batch are:-",(j + 1));

pq.enqueue(bottles[0 + k]);

pq.enqueue(bottles[1 + k]);

pq.enqueue(bottles[2 + k]);

pq.enqueue(bottles[3 + k]);

pq.enqueue(bottles[4 + k]);

pq.dequeue();

pq.dequeue();

pq.dequeue();

pq.dequeue();

pq.dequeue();

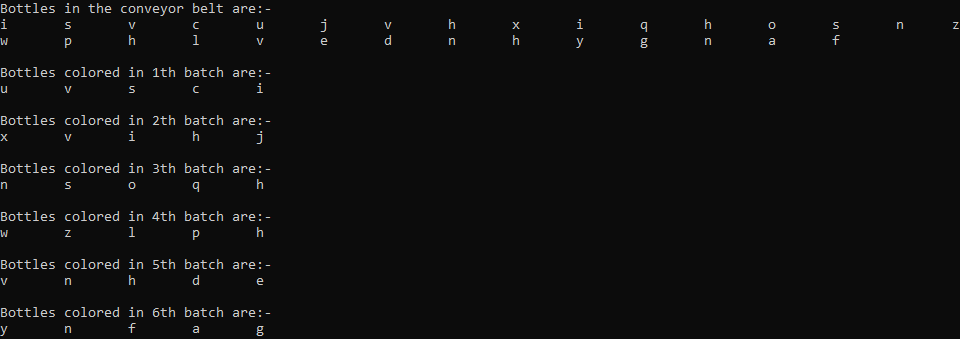
k += 5;

}

bottles = null;

}

**Output:-**

****

**Question 2:-**

**Solution:-**

static void printSubset(int[] elements,int count)

{

int[] positions = new int[] {0,1,2,3};

while (positions[0] <= count-4)

{

if (positions[3] < count)

{

Console.Write("{0,-10} {1,-10} {2,-10} {3,-10}\n", elements[positions[0]], elements[positions[1]], elements[positions[2]], elements[positions[3]]);

positions[3]++;

}

else if(positions[3] == count) {

positions[2]++;

positions[3] = positions[2] + 1;

}

else if (positions[2] == count)

{

positions[1]++;

positions[2] = positions[1] + 1;

positions[3] = positions[1] + 2;

}

else if(positions[1] == count)

{

positions[0]++;

positions[1] = positions[0] + 1;

positions[2] = positions[0] + 2;

positions[3] = positions[0] + 3;

}

}

}

static void Main(string[] args)

{

Random r = new Random();

SortedSet<int> set = new SortedSet<int>();

int[] arr = new int[10];

while(set.Count < 5)

{

set.Add(r.Next(1, 101));

}

int i = 0;

Console.WriteLine("Original array:-");

foreach(int j in set)

{

arr[i] = j;

Console.Write("{0,-10} ",arr[i]);

i++;

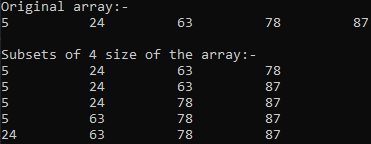
}

Console.WriteLine("\n\nSubsets of 4 size of the array:-");

printSubset(arr,set.Count);

}

**Output:-**

****

**Question 3:-**

**Solution:-**

static void sortedMerge(string[] array1,string[] array2)

{

Array.Sort(array1);

Array.Sort(array2);

string[] temp = new string[10];

int i = 0, j = 0,k = 0;

while(i < 5 && j < 5)

{

if(String.Compare(array1[i],array2[j]) < 0)

{

temp[k] = array1[i];

i++;

k++;

}

else if(String.Compare(array1[i], array2[j]) == 0)

{

temp[k] = array1[i];

temp[k+1] = array1[j];

i++;

j++;

k += 2;

}

else

{

temp[k] = array2[j];

j++;

k++;

}

}

for(k = i < 5 ? i + 5: j + 5;k<10;k++)

{

if (i < 5) {

temp[k] = array1[k-5];

}

else

{

temp[k] = array2[k-5];

}

}

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

{

Console.WriteLine(temp[l]);

}

}

static void Main(string[] args)

{

String[] array1 = new String[5];

String[] array2 = new String[5];

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

{

Console.WriteLine("Enter {0}th name",i);

if (i < 5)

{

array1[i] = Console.ReadLine();

}

else

{

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

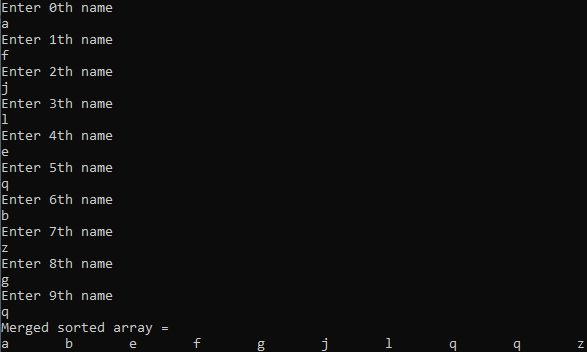
}

}

sortedMerge(array1, array2);

}

**Output:-**

****