|  |
| --- |
| namespace \_08.\_Anonymous\_Threat  {  internal class Program  {  static void Main(string[] args)  {  List<string> data = Console  .ReadLine()  .Split()  .ToList();  string input;  while ((input = Console.ReadLine()) != "3:1")  {  string[] command = input.Split();  switch (command[0])  {  case "merge":  int startIndex = int.Parse(command[1]);  int endIndex = int.Parse(command[2]);  Merging(data, startIndex, endIndex);  break;  case "divide":  int index = int.Parse(command[1]);  int partitions = int.Parse(command[2]);  Dividing(data, index, partitions);  break;  }  }  Console.WriteLine(string.Join(" ", data));  }  static void Merging(List<string> data, int startIndex, int endIndex)  {  if (startIndex < 0)  {  startIndex = 0;  }  else if (startIndex > data.Count - 1)  {  startIndex = data.Count - 1;  }  if (endIndex > data.Count)  {  endIndex = data.Count - 1;  }  else if (endIndex > data.Count - 1)  {  endIndex = data.Count - 1;  }  List<string> temp = new List<string>();  for (int i = startIndex; i <= endIndex; i++)  {  temp.Add(data[i]);  }  data[startIndex] = string.Join("", temp);  for (int i = startIndex + 1; i <= endIndex; i++)  {  data.RemoveAt(startIndex + 1);  }  }  static void Dividing(List<string> data, int index, int partitions)  {  List<string> temp = new List<string>();  string dividingNumber = data[index];  int length = dividingNumber.Length / partitions;  int addLength = dividingNumber.Length % partitions;  for (int i = 0; i < partitions; i++)  {  if (i == partitions - 1)  {  length += addLength;  }  temp.Add(dividingNumber.Substring(0, length));  dividingNumber = dividingNumber.Remove(0, length);  }  data.RemoveAt(index);  data.InsertRange(index, temp);  }  }  } |