DAY 1

Date:04/08/25
Name: Nancy M

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Diagnostics;

namespace Sample
{
    class Program
    {

        static int sum_amount(List<sales_report> salesreport)
        {
            int sum = 0;
            foreach (var report in salesreport)
            {
                sum = sum + report.cos_purchase;
            }
            return sum;
        }


        static Tuple<int, int> GetMaxMinPurchase(List<sales_report> reportList)
        {
            int max = reportList.Max(s => s.cos_purchase);
            int min = reportList.Min(s => s.cos_purchase);
            return Tuple.Create(max, min);
        }


        public enum shop_name
        {
            max,
            unlimited,
            trends,
        }
        struct sales_report
        {
            public string cos_name;
            public string cos_place;
            public int cos_purchase;
            public string cos_brand;

            public sales_report(string _cos_name, string _cos_place, int _cos_purchase, string
_cos_brand)
            {
                cos_name = _cos_name;
                cos_place = _cos_place;
                cos_purchase = _cos_purchase;
                cos_brand = _cos_brand;
            }
        }

        static void Main(string[] args)
        {
            sales_report s1 = new sales_report("nancy", "cbe", 2000, "max");
            sales_report s2 = new sales_report("jency", "ooty", 3000, "unlimited");
            sales_report s3 = new sales_report("maria", "chennai", 4000, "trends");

            List<sales_report> salesreport = new List<sales_report>();
            salesreport.Add(s1);
            salesreport.Add(s2);
            salesreport.Add(s3);

            foreach (sales_report report in salesreport)
            {
                Debug.WriteLine(string.Format("{0}, {1}, {2}, {3}", report.cos_name, report.cos_place,
report.cos_purchase, report.cos_brand));
            }

            Debug.WriteLine(string.Format("Total Purchase Amount: {0}",
sum_amount(salesreport)));

            Func<List<sales_report>, int> totcos =
                (List<sales_report> reportList) => reportList.Count;

            Debug.WriteLine(string.Format("Total count is: {0}", totcos(salesreport)));

            Func<List<sales_report>, IOrderedEnumerable<sales_report>> ascorder =
                reportList => reportList.OrderBy(d => d.cos_purchase);
            Debug.WriteLine("Ascending order:");
            foreach (sales_report asc in ascorder(salesreport))
            {
                Debug.WriteLine(asc.cos_purchase);
            }

            Func<List<sales_report>, IOrderedEnumerable<sales_report>> dscorder =
                reportList => reportList.OrderByDescending(d => d.cos_purchase);
            Debug.WriteLine("Descending order:");
            foreach (sales_report dsc in dscorder(salesreport))
            {
                Debug.WriteLine(dsc.cos_purchase);
            }

            var maximumvalue = salesreport.Max(a => a.cos_purchase);
            Debug.WriteLine("Maximum value: " + maximumvalue);

            var minimumvalue = salesreport.Min(b => b.cos_purchase);
            Debug.WriteLine("Minimum value: " + minimumvalue);

            var sumvalue = salesreport.Sum(c => c.cos_purchase);
            Debug.WriteLine("Sum using anonymous function: " + sumvalue);


            var myTuple = Tuple.Create("Nirmal", 409);
            Debug.WriteLine(myTuple.Item1);


            var result = GetMaxMinPurchase(salesreport);
            Debug.WriteLine(string.Format("Tuple function - Min: {0} and Max: {1}", result.Item2,
result.Item1));
        }
    }
}
```

Output:

```
nancy, cbe, 2000, max
jency, ooty, 3000, unlimited
maria, chennai, 4000, trends
Total Purchase Amount: 9000
Total count is: 3
```

```
Ascending order:
2000
3000
4000
Descending order:
4000
3000
2000
Maximum value: 4000
Minimum value: 2000
Sum using anonymous function: 9000
Nirmal
Tuple function - Min: 2000 and Max: 4000
```