Exercise No: 1A

What is Data Pre-processing?

Data pre-processing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behavior's or trends, and is likely to contain many errors. Data pre-processing is a proven method ofresolving such issues. Data pre- processing prepares raw data for further processing. Data preprocessing is used database-driven applications such ascustomer relationship management and rule- based applications (like neuralnetworks).

What is Data Smoothing?

Data smoothing is done by using an algorithm to remove noise from a data set. This allows important patterns to stand out. Data smoothing can be usedto help predict trends, such as those found in securities prices. The idea behind data smoothing is that it can identify simplified changes in order to help predict different trends and patterns. It acts as an aid for statisticians or traders who need to look at a lot of data that can often be complicated to digest to find patterns they would not otherwise see.

What is Smoothing by Bin Means?

In smoothing by bin means, each value in a bin is replaced by the mean value of the bin.

Example:

Perform data cleaning technique using smoothing by BIN MEANS on elements 4, 8, 15, 21, 21, 24, 25, 34, 28 whose bin size is 3.

Sort the elements and form a matrix using bin size.

4 8 15

21 21 24

25 28 34

Calculate mean value for

each bin. Bin 1: 4 8

15

Bin 1 Mean: sum of elements of bin 1/bin size

= (4+8+15)/3

= 9

Bin 2: 21 21 24

Bin 2 Mean: sum of elements of bin 2/bin size

= (21+21+24)/3

= 22

Bin 3: 25 28 34

Bin 3 Mean: sum of elements of bin 3/bin size

= (25+28+34)/3

= 29

Mean Values:

9 9 9

22 22 22

29 29 29

AIM:

Write a C program to perform data cleaning techniques using smoothing by binmeans.

SOURCE CODE:

```
#include<stdio.h>
void main()
{
        int i, k, n, bin, a [50], temp, sum;
        float mean;
        printf("Enter no of bins:");
        scanf("%d",&bin);
        printf("Enter no of values to be enter in each bin:");
        scanf("%d",&n);
        printf("Enter elements\n");
        for(i=0;i<n*bin;i++)
        {
                scanf("%d",&a[i]);
        }
        printf("The given elements are\n");
        for(i=0;i<n*bin;i++)
        {
                for(k=0;k<n*bin;k++)
                {
                        if(a[i]<a[k])
                         {
                                temp=a[i];
                                a[i]=a[k];
                                a[k]=temp;
                         }
                }
        }
        for(i=0;i<bin;i++)
        {
                for(k=0;k<n;k++)
                {
                        printf("%2d\t",a[k+i*n]);
                }
                printf("\n");
```

```
        }
        printf("The mean values are\n");
            for(i=0;i<bin;i++)
             {
                  mean=0,sum=0;
                  for(k=0;k<n;k++)
                   {
                      sum=sum+a[k+i*n];
                   }
                   mean=(float)sum/n;
                    for(k=0;k<n;k++)
                    {
                         a[k+i*n]=mean;
                         printf("%f\t",mean);
                    }
                    printf("\n");
             }
    }
```

OUTPUT:

```
Enter no. of bins: 3
Enter each bin size: 3
Enter elements:
11 12 13 14 15 16 17 18 19
Elements are:
19    18    17
16    15    14
13    12    11
Mean values are:
18.000000   18.000000   18.000000
15.000000   15.000000   15.000000
12.000000   12.000000   12.000000
```

Exercise No:1B

What is Smoothing by Bin Medians?

In Smoothing by Bin Medians, each value in a bin is replaced by the median value of the bin.

Example:

Perform data cleaning technique using smoothing by BIN MEDIANS on elements 4, 8, 15, 21, 21, 24, 25, 34, 28 whose bin size is 3.

Sort the elements and form a matrix using bin size.

4 8 15

21 21 24

25 28 34

Calculate median value for each bin.

Bin 1: 4 8 15

Bin 1 Median: Middle value of the bin = 8

Bin 2: 21 21 24

Bin 2 Median: Middle value of the bin = 21

Bin 3: 25 28 34

Bin 3 Median: Middle value of the bin = 28

Median Values:

8 8 8

21 21 21

28 28 28

AIM:

Write a C program to perform data cleaning techniques using smoothing by bin medians.

SOURCE CODE:

```
#include<stdio.h>
void main()
{
        int i,k,n,bin,a[50],temp,sum;
        float median;
        printf("Enter no of bins:");
        scanf("%d",&bin);
        printf("Enter no of values to be enter in each bin:");
        scanf("%d",&n);
        printf("Enter elements\n");
        for(i=0;i<n*bin;i++)
        {
                scanf("%d",&a[i]);
        }
        printf("The given elements are\n");
        for(i=0;i<n*bin;i++)
        {
        for(k=0;k<n*bin;k++)
        {
                if(a[i]<a[k])
                {
                   temp=a[i];
                   a[i]=a[k];
                   a[k]=temp;
                }
        }
       }
       for(i=0;i<bin;i++)
       {
        for(k=0;k<n;k++)

        {
```

```
            printf("%2d\t",a[k+i*n]);
          }
          printf("\n");
        }
  printf("The median values are\n");
  for(i=0;i<bin;i++)
   {
        if(n%2==0)
        {
              median=(float)(a[n/2-1+i*n]+a[n/2+i*n])/2;
        }
       else
       {
            median=a[n/2+i*n];
       }
       for(k=0;k<n;k++)
       {
            a[k+i*n]=median;
            printf("%f\t",median);
       }
       printf("\n");
}
}
```

OUTPUT:

```
Enter no. of bins: 3
Enter each bin size: 3
Enter elements:
20 21 22 17 16 15 14 13 12
Elements are:
22    21    20
17    16    15
14    13    12
Median values:
21.000000   21.000000   21.000000
16.000000   16.000000   16.000000
13.000000   13.000000   13.000000
```

Exercise No:1C

What is Smoothing by Bin Boundaries?

In Smoothing by Bin Boundaries, the minimum and maximum values in a given bin are identified as the bin boundaries.

Example:

Perform data cleaning technique using smoothing by BIN BOUNDARIES on elements 4, 8, 15, 21, 21, 24, 25, 34, 28 whose bin size is 3.

Sort the elements and form a matrix using bin size.

4 8 15

21 21 24

25 28 34

Calculate boundary value for each element in the bins by finding difference of that element with lower and upper boundary values. The boundary value which gives minimum difference will be replaced in that element place in the bin.

Bin 1: 4 8 15

Bin 1:

Min=4 and Max=15

For value 4, (4-4) < (15-4) so replace 4 with lower boundary value 4.

For value 8, (8-4) < (15-8) so replace 8 with lower boundary value 4.

For value 15, (15-4) > (15-15) so replace 15 with upper boundary value 15.

Bin 2: 21 21 24

Bin 2:

Min=21 and Max=24

For value 21, (21-21) < (24-21) so replace 21 with lower boundary value 21.

For value 21, (21-21) < (24-21) so replace 21 with lower boundary value 21.

For value 24, (24-21) > (24-24) so replace 24 with upper boundary value 24.

Bin 3: 25 28 34

Bin 3:

Min=25 and Max=34

For value 25, (25-25) < (34-25) so replace 25 with lower boundary value 25.

For value 28, (28-25) < (34-28) so replace 28 with lower boundary value 25.

For value 34, (34-25) > (34-34) so replace 34 with upper boundary value 34.

Bin Boundaries:

4 4 15

21 21 24

25 25 34

AIM:

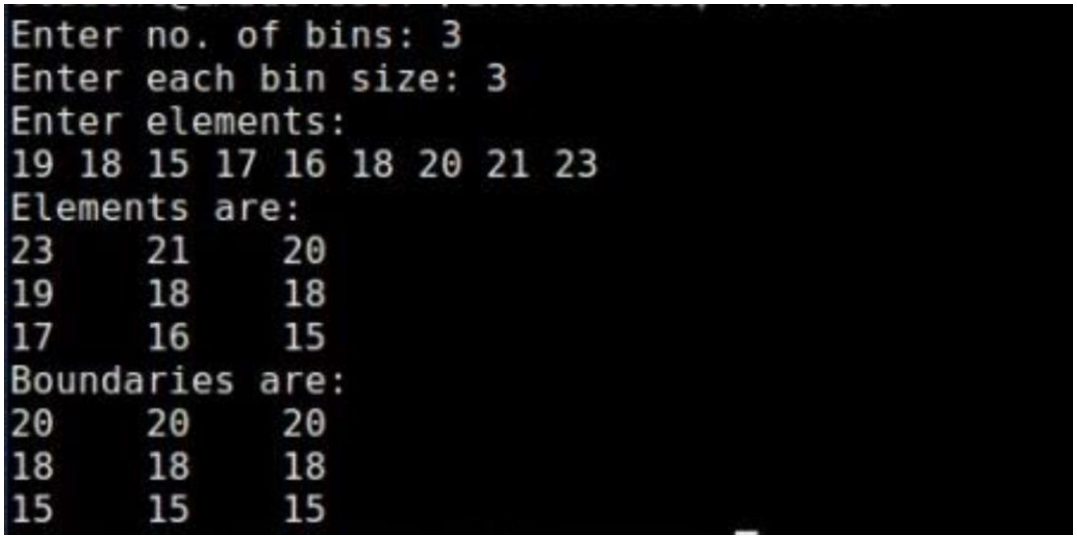
Write a C program to perform data cleaning techniques using smoothing by bin boundaries.

SOURCE CODE:

```
#include<stdio.h>
void main()
{
        int i,j,n,nb,a[50],temp,sum,mid,b[10][10];
        printf("Enter no of bins:");
        scanf("%d",&n);
        printf("Enter values each bin:");
        scanf("%d",&nb);
        printf("Enter values\n");
        for(i=0;i<n*nb;i++)
        {
           scanf("%d",&a[i]);
        }
        printf("The given values\n");
        for(i=0;i<n*nb;i++)
        {
           for(j=0;j<n*nb;j++)
                {
                        if(a[i]<a[j])
                        {
                                temp=a[i];
                                a[i]=a[j];
                                a[j]=temp;
                        }
                }
        }
        for(i=0;i<n;i++)
        {
          for(j=0;j<nb;j++)
           {
```

```
                printf("%2d\t",a[j+i*nb]);
                b[i][j]=a[j+i*nb];
            }
          printf("\n");
      }
      printf("boundaries are\n");
      for(i=0;i<n;i++)
      {
         sum=b[i][0]+b[i][nb-1];
         mid=sum/2;
         for(j=0;j<nb;j++)
         {
                        if(b[i][j]<=mid)
                        b[i][j]=b[i][0];
                        else
                        b[i][j]=b[i][n-1];
                        printf("%d\t",b[i][j]);
          }
         printf("\n");
          }
}
```

OUTPUT:



Exercise No: 2A

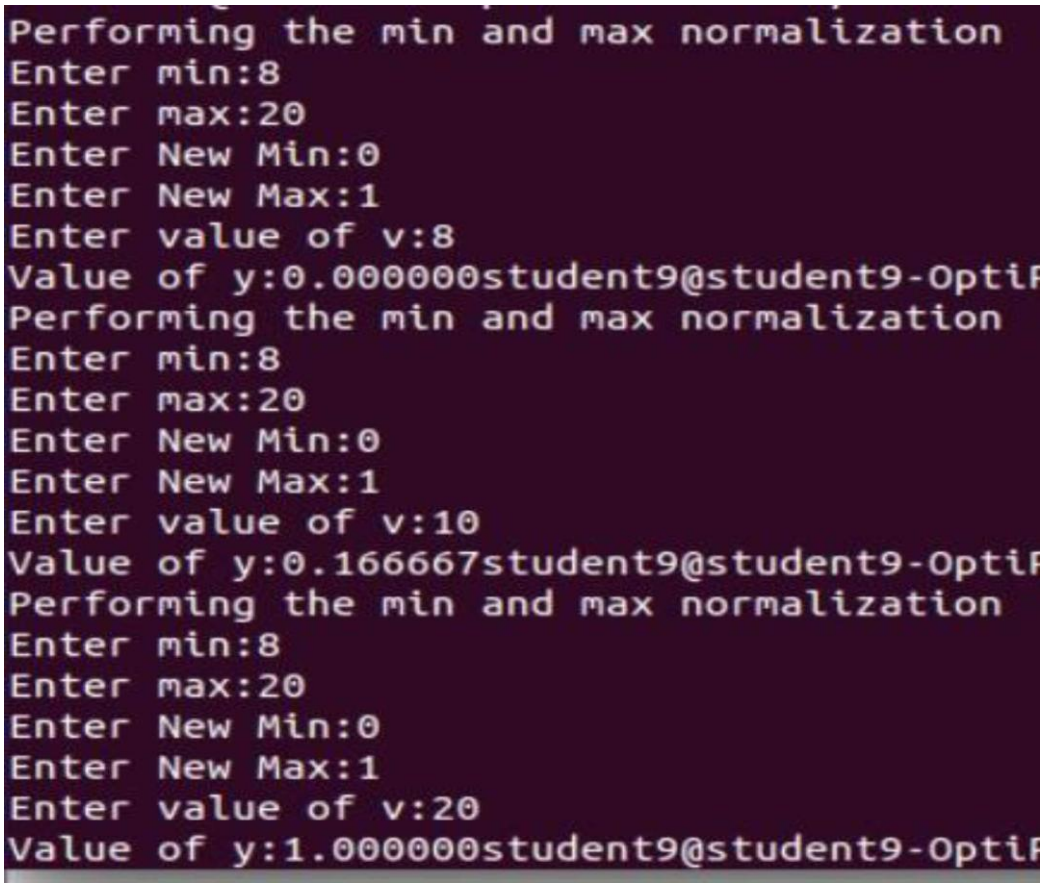
Aim:

Write a C program to perform Data Transformation Technique using Min-Max normalization.

Source Code:

```
#include<stdio.h>
void main()
{
        float  min,max,newmin,newmax,y,v;
        printf("Performing the min and max normalization\n");
        printf("Enter min:");
        scanf("%f",&min);
        printf("Enter max:");
        scanf("%f",&max);
        printf("Enter New Min:");
        scanf("%f",&newmin);
        printf("Enter New Max:");
        ("%f",&newmax);
        printf("Enter value of v:");
        scanf("%f",&v);
        y=((v-min)/(max-min))*(newmax-newmin)+newmin;
        printf("Value of y:%f",y);
}
```

OUTPUT:


Performing the min and max normalization
Enter min:8
Enter max:20
Enter New Min:0
Enter New Max:1
Enter value of v:8
Value of y:0.000000student9@student9-Optif
Performing the min and max normalization
Enter min:8
Enter max:20
Enter New Min:0
Enter New Max:1
Enter value of v:10
Value of y:0.166667student9@student9-Optif
Performing the min and max normalization
Enter min:8
Enter max:20
Enter New Min:0
Enter New Max:1
Enter value of v:20
Value of y:1.000000student9@student9-Optif

Exercise No: 2B

Aim:

Write a C program to perform Data Transformation Technique using Z-score normalization.

Source Code:

```
#include<stdio.h>
#include<math.h>
int main()
{
    int i,n;
    float v,v1,sig,avg,sum=0,a[20];
    printf("Enter number of elements to be entered\n");
    scanf("%d",&n);
    printf("Enter the elements\n");
    for(i=0;i<n;i++)
    scanf("%f",&a[i]);
    printf("enter v value\n");
    scanf("%f",&v);
    for(i=0;i<n;i++)
    sum=sum+a[i];
    avg=sum/n;
    printf("sum is =%f\n",sum);
```

```
        printf("Avg =%f\n",avg);
        sum=0;
        for(i=0;i<n;i++)
        {
                sum=sum+pow(avg-a[i],2);
        }
        sig=sum/n;
        printf("varience is %f\n",sqrt(sig));
        v1=(v-avg)/sqrt(sig);
        printf("v1 value=%f\n",v1);
}
```

OUTPUT:

```
Enter no of values to be entered
5
Enter the elements
100 200 400 900 1000
enter v value
400
sum is=2600.000000
avg=520.000000
Variance is 365.513338
v1 value=-0.328305
```

Exercise No: 2C

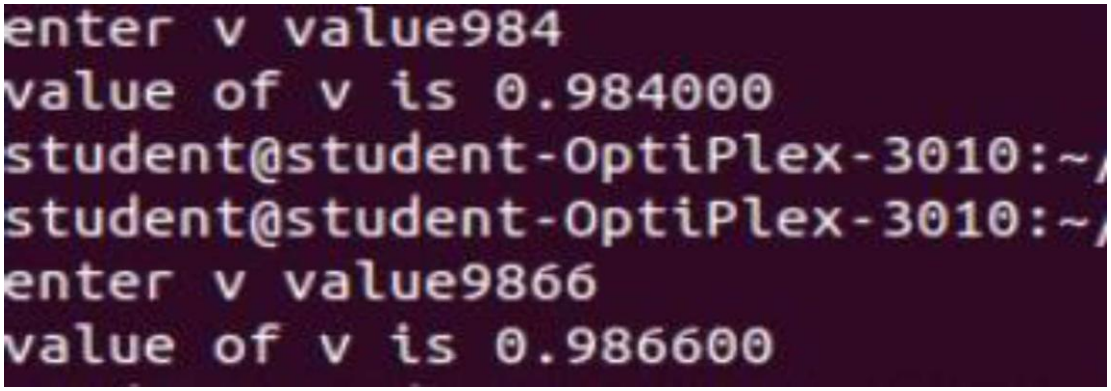
Aim:

Write a C program to perform Data Transformation Technique using Normalization by decimal scaling.

Source Code:

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
int main()
{
        float vd=2;
        int j=1,v;
        printf("enter v value:");
        scanf("%d",&v);
        v=abs(v);
        while(vd>=1)
        {
                vd=v/pow(10,j);
                j++;
        }
        printf("value of v' is %f\n",vd);
}
```

OUTPUT:


enter v value984
value of v is 0.984000
student@student-OptiPlex-3010:~
student@student-OptiPlex-3010:~
enter v value9866
value of v is 0.986600

Exercise No: 3

Aim:

Write a C program to implement Apriori algorithm.

Source Code:

```
#include<stdio.h>
static int row,col,min_freq=2;
void findFrequentSets(int items[][100],int transactions[][100],intno_of_items[],int
no_of_transactions,int result_item[][100]);
void generateSubsets(int items[][100],int setCount,int result_item[][100]);
void main()
{
        int  no_of_transactions,s=0,k=1,l=0,i,j,setCount=2,maxCount;
        int transactions[100][100],no_of_items[100],items[100][100];
        int result_item[100][100];
        printf("enter no.of transactions\n");
        scanf("%d",&no_of_transactions);
        for(i=1;i<=no_of_transactions;i++)
        {
           printf("enter no.of items in transaction:%d\n",i);
           scanf("%d",&no_of_items[i]);
            printf("enter %d items for transaction:%d\n",no_of_items[i],i);
            for(j=1;j<=no_of_items[i];j++)
            scanf("%d",&transactions[i][j]);
        }
       k=0;
        for(i=1;i<=no_of_transactions;i++)
        {
            for(j=1;j<=no_of_items[i];j++)
                    {
                            for(l=1;l<=k;l++)
                            if(items[l][1]==transactions[i][j])
                            break;
                            if(l>k)
                            items[++k][1] = transactions[i][j];
                    }
        }
```

```
                row=k;
                col=1;
                maxCount=k;

findFrequentSets(items,transactions,no_of_items,no_of_transactions,res ult_item);
            printf("\n");
            for(i=1;i<=row;i++)
              {
                for(j=1;j<=col;j++)
                 printf("%d ",result_item[i][j]);
                printf("\n");
             }
            while(setCount<=maxCount)
            {
               generateSubsets(result_item,setCount,items);

findFrequentSets(items,transactions,no_of_items,no_of_transactions,res ult_item);
            for(i=1;i<=row;i++)
            {
              for(j=1;j<=col;j++)
              printf("%d ",result_item[i][j]);
              printf("\n");
            }
            setCount++;
            }
 }
 void findFrequentSets(int items[][100],int transactions[][100],int
 no_of_items[],int no_of_transactions,int result_item[][100])
 {
       int i,j,k,l,p,q,m,count;
        k=1;l=1;p=1;q=1;
            while(l<=row)
              {
                 count=0;
            for(i=1;i<=no_of_transactions;i++)
            {
               m=1;
               for(j=1;j<=no_of_items[i];j++)
                {
                  if(m<=col)
```

```
                {
                  if(transactions[i][j]==items[l][m])
                    {
                            m++;
                            j=0;
                     }
                }
                else
                        break;
        if(m>col)

        ++count;
  }
  if(count>=min_freq)
  {
        q=1;
        for(m=1;m<=col;m++)
        result_item[p][q++]=items[l][m];
        p++;
}
        l++;
}
        for(i=1;i<=row;i++)
        for(j=1;j<=col;j++)
        items[i][j]=0;
        row=p-1;col=q-1;
  }
  void generateSubsets(int items[][100],int setCount,int result_item[][100])
  {
        int i=1,j,k,l,count=0,newRow=0,newCol=0,p,q,m,n,r=1,b[row+1];
        while(i<=row)
        {
           j=i;
           k=1;
           while(j<=row)
            {
              if(j==row)
                 {
                 if(count+1==2)
                   {
```

```
            b[k++]=j;
            newRow++;
            newCol=1;
            for(l=1;l<k;l++)
            {
               for(p=1;p<=col;p++)
                {
                   for(q=1;q<newCol;q++)
                   if(result_item[newRow][q]==items[b[l]][p])
                   break;
                   if(q>=newCol)
                 result_item[newRow][newCol++]=items[b[l]][p];
                }
           }
            if(newCol-1!=setCount)
            {
                for(m=1;m<newCol;m++)
                result_item[newRow][m]=0;
                newRow--;
            }
           else
            {
               for(m=1;m<newRow;m++)
            {
            r=1;
            for(n=1;n<newCol;n++)
            {
               if(result_item[m][n]==result_item[newRow][r])
               {
                 n=1;
                  r++;
                  if(r>setCount)
                  {
                      for(p=1;p<newCol;p++)
                      result_item[newRow][p]=0;
                      newRow--;
                      break;
                  }
        }
            if(n<=setCount)
```

```
        break;
        }
    }
    k--;
    b[k]=0;
        }
        if(k-1<=1)
          {
                count=0;
                for(l=1;l<k;l++)
                b[l]=0;
                break;
          }
          k--;
          count--;
          j=b[k]+1;
          b[k]=0;
      }
        else if(count+1<2)
      {
           count++;
           b[k++]=j;
           j++;
        }
        else if(count+1==2)
        {
          b[k++]=j;
          newRow++;
          newCol=1;
          for(l=1;l<k;l++)
           {
                for(p=1;p<=col;p++)
                {
                  for(q=1;q<newCol;q++)
                  if(result_item[newRow][q]==items[b[l]][p])
                  break;
                  if(q>=newCol)
                  result_item[newRow][newCol++]=items[b[l]][p];
                }
        }
```

```
            if(newCol-1!=setCount)
            {
              for(m=1;m<newCol;m++)
              result_item[newRow][m]=0;
              newRow--;
            }
            else
            {
               for(m=1;m<newRow;m++)
               {
                  r=1;
                for(n=1;n<newCol;n++)
                {
                      if(result_item[m][n]==result_item[newRow][r])
                      {
                        n=1;
                         r++;
                         if(r>setCount)
                          {
                            for(p=1;p<newCol;p++)
                             result_item[newRow][p]=0;
                                 newRow--;
                               break;
                               }
                       }
                }
   if(n<=setCount)
    break;
  }
 }
    k--;
   b[k]=0;
    j++;
  }
     else
     j++;
 }
 i++;
 }
 row=newRow;
 col=newCol-1;
 }
```

OUTPUT:

```
enter no.of transactions
9
enter no.of items in transaction:1
2
enter 2 items for transaction:1
2 4
enter no.of items in transaction:2
2
enter 2 items for transaction:2
2 3
enter no.of items in transaction:3
3
enter 3 items for transaction:3
1 2 4
enter no.of items in transaction:4
2
enter 2 items for transaction:4
1 3
enter no.of items in transaction:5
2
enter 2 items for transaction:5
2 3
enter no.of items in transaction:6
2
enter 2 items for transaction:6
2 3
enter no.of items in transaction:7
2
enter 2 items for transaction:7
1 3
enter no.of items in transaction:8
4
enter 4 items for transaction:8
```

```
student@student-OptiPlex-3010: ~/17091A05C5
2
enter 2 items for transaction:4
1 3
enter no.of items in transaction:5
2
enter 2 items for transaction:5
2 3
enter no.of items in transaction:6
2
enter 2 items for transaction:6
2 3
enter no.of items in transaction:7
2
enter 2 items for transaction:7
1 3
enter no.of items in transaction:8
4
enter 4 items for transaction:8
1 2 3 5
enter no.of items in transaction:9
3
enter 3 items for transaction:9
1 2 3

2
4
3
1
2 4
2 3
2 1
3 1
2 3 1
```

Exercise No: 4A

Aim:

Write a C program to implement Bayes classification technique.

Source Code:

```
#include<stdio.h>

#include<string.h>

void class(int ,int);

char cls[10][20],titems[50][20][20],attr[10][20];

int pcount[20],count[10],fc=0,c=0;

float p[10],prob[20],pre[10],result[10];

int main()

{
        char tup[15][20];
        int i,j,n,tuples,k,ans=0,t=0;
        printf("enter no of attributes:");
        scanf("%d",&n);
        printf("enter no of tuples:");
        scanf("%d",&tuples);
        printf("enter %d attributes\n",n);
        for(i=0;i<n;i++)
        scanf("%s",attr[i]);
        for(i=0;i<tuples;i++)
        {

        printf("enter tuple%d\n",i+1);
        for(j=0;j<n;j++)
        scanf("%s",titems[i][j]);
      }

      printf("enter test tuple\n");
      for(i=0;i<n-1;i++)
```

```
        scanf("%s",tup[i]);
        class(n,tuples);
        for(i=0;i<fc;i++)
        p[i]=count[i]/(float)tuples;
    for(i=0;i<fc;i++)
    {
          for(j=1;j<n-1;j++)
          {
              pcount[j]=0;
              for(k=0;k<tuples;k++)
              {
                  if(strcmp(titems[k][j],tup[j])==0 &&strcmp(cls[i],titems[k][n-
                  1])==0) pcount[j]+=1;
              }
              if(pcount[j]!=0 && t==0)
               prob[c++]=pcount[j]/(float)count[i];
               else
               {
                 t=1;
                 prob[c++]=(pcount[j]+1)/(float)count[i];
                 }
    }
  }
  j=0;
  for(i=0;i<fc;i++)
  {
      pre[i]=1.0;
      for( ;j<((i+1)*(c/fc));j++)
      pre[i]*=prob[j];
   }
  for(i=0;i<fc;i++)
  {
     result[i]=pre[i]*p[i];
     if(i>0 && result[i]>result[i-1])
      ans=i;
  }
  printf("The test tuple belongs to %s class",cls[ans]);
  }
```

```
void class(int p,int q)
 {
      int i=0,k,t=0;

      strcpy(cls[fc++],titems[0][p-1]);

      for(i=1;i<q;i++)

      {
          t=0;
           for(k=0;k<fc;k++)
            {
              if(strcmp(titems[i][p-1],cls[k])==0)
              {
                t=-1;
                break;
              }
      }
       if(t!=-1)
        strcpy(cls[fc++],titems[i][p-1]);
 }
   for(i=0;i<fc;i++)
   {
      count[i]=0;
       for(k=0;k<q;k++)
        {
          if(strcmp(titems[k][p-1],cls[i])==0)
           count[i]+=1;
      }

}
```

OUTPUT:


enter no of attributes:4
enter no of tuples:10
enter 4 attributes
color type origin class
enter tuple1
red sports dom yes
enter tuple2
red sports dom no
enter tuple3
red sports dom yes
enter tuple4
yellow sports dom no
enter tuple5
yellow sports imp no
enter tuple6
yellow suv imp no
enter tuple7
yellow suv imp yes
enter tuple8
yellow suv dom no
enter tuple9
red suv imp no
enter tuple10
red sports imp yes
enter test tuple
yellow sports imp
The test tuple belongs to yes class

Exercise No: 4B

Aim:

Write a C program to implement Nearest Neighbor Classification technique

Source Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
int  tuples,attr;
float near[20];
void NNC(int d[20][20],int t[1][20]) ;
int isNear()
int main()
{
        int i,j,k,data[20][20],test[1][20],s;
        printf("enter no of tuples:");
        scanf("%d",&tuples);
        printf("enter no of attributes:");
        scanf("%d",&attr);
        printf("enter %d tuples\n",tuples);
        for(i=0;i<tuples;i++)
        {
             for(j=0;j<attr;j++)
            {
            scanf("%d",&data[i][j]);
            }
         }
printf("enter test tuple\n");
```

```
for(i=0;i<attr;i++)

{
scanf("%d",&test[0][i]);

}

printf("enter k value:");

scanf("%d",&k);

NNC(data,test);

printf("The nearest neighbours are\n");

for(i=0;i<k;i++)

{
    s=isNear();

    printf("(");

    for(j=0;j<attr;j++)

     {
        printf("%3d",data[s][j]);
     }
      printf(")\n");
   }
}
void NNC(int d[][20],int t[][20])
{
        int i,j,sum=0;

        for(i=0;i<tuples;i++)

        {

         for(j=0;j<attr;j++)

         {
            sum+=((d[i][j]-t[0][j])*(d[i][j]-t[0][j]));

         }

        near[i]=sqrt(sum);
```

```
            sum=0;
        }
    }
    int isNear()
    {
       float t=near[0];
        int s=0,i;
        for(i=1;i<tuples;i++)
        {
            if(t>=near[i] && near[i]!=999)
            {
                t=near[i];
                  s=i;
           }
    }
    near[s]=999;
    return s;
    }
```

OUTPUT:

```
enter no of tuples:5
enter no of attributes:2
enter 5 tuples
4 3
6 7
7 8
5 5
8 8
enter test tuple
6 8
enter k value:3
The nearest neighbours are
(  7  8)
(  6  7)
(  8  8)
```

Exercise No: 5

Aim:

Write a C program to implement k-means clustering algorithm.

Source Code:

```
#include<stdio.h>
void main()
{
        int i1,i2,i3,t1,t2,m1,m2,om1,om2;
        int k0[10],k1[10],k2[10];
        printf("Enter 10 numbers:");
        for(i1=0;i1<10;i1++)
        {
                scanf("%d",&k0[i1]);
        }
        printf("Enter intial mean 1:");
        scanf("%d",&m1);
        printf("Enter intial mean 2:");
        scanf("%d",&m2);
        do
        {
            om1=m1;
            om2=m2;
            i1= i2=i3=0;
           for(i1=0;i1<10;i1++)
            {
                t1=k0[i1]-m1;
```

```
                if(t1<0)
                {
                  t1=-t1;
                }
                t2=k0[i1]-m2;
                if(t2<0)
                {
                  t2=-t2;
                }
                if(t1<t2)
                {
                  k1[i2]=k0[i1];
                   i2++;
                }
                else
                {
                   k2[i3]=k0[i1];
                   i3++;
                   }
      }
   t2=0;
   for(t1=0;t1<i2;t1++)
   {
          t2=t2+k1[t1];
   }
```

```
m1=t2/i2;

t2=0;

for(t1=0;t1<i3;t1++)

{
        t2=t2+k2[t1];
}
m2=t2/i3;

printf("\nCluster 1:");

for(t1=0;t1<i2;t1++)

{
        printf("%d ",k1[t1]);
}

printf("\nm1=%d",m1);

printf("\nCluster 2:");

for(t1=0;t1<i3;t1++)

{
        printf("%d ",k2[t1]);
}

printf("\nm2=%d",m2);

printf("\n -------- ");

}
while(m1!=om1 && m2!=om2);

printf("\nClusters created");

}
```

OUTPUT:

```
Enter 10 numbers:2 4 10 12 3 20 30 11 25 23
Enter intial mean 1:2
Enter intial mean 2:16

Cluster 1:2 4 3
m1=3
Cluster 2:10 12 20 30 11 25 23
m2=18
---------
Cluster 1:2 4 10 3
m1=4
Cluster 2:12 20 30 11 25 23
m2=20
---------
Cluster 1:2 4 10 3 11
m1=6
Cluster 2:12 20 30 25 23
m2=22
---------
Cluster 1:2 4 10 12 3 11
m1=7
Cluster 2:20 30 25 23
m2=24
---------
Cluster 1:2 4 10 12 3 11
m1=7
Cluster 2:20 30 25 23
m2=24
---------
Clusters createdstudent@student-OptiPlex-3010:
```