**Name: Amaan Shaikh Roll no: 35 SYCSE-A**

FDS Experiment no 1-4 Codes

**Experiment no 1 Code:**

*//Program Matric Operations*

#include <stdio.h>

#include <stdlib.h>

void accept(int X[10][10], int a, int b)

{

    int i, j;

    for(i=0;i<a;i++)

    {

        for(j=0;j<b;j++)

        {

            printf("Enter element [%d%d] :\t", i, j);

            scanf("%d", &X[i][j]);

        }

    }

}

void display(int Z[10][10], int a, int b)

{

    int i, j;

    for(i=0;i<a;i++)

    {

        for(j=0;j<b;j++)

        {

            printf(" %d \t", Z[i][j]);

        }

        printf("\n");

    }

}

void add(int X[10][10], int Y[10][10], int Z[10][10], int m, int n)

{

    int i, j;

    for(i=0;i<m;i++)

        for(j=0;j<n;j++)

            Z[i][j] = X[i][j] + Y[i][j];

}

void subtract(int X[10][10], int Y[10][10], int Z[10][10], int m, int n)

{

    int i, j;

    for(i=0;i<m;i++)

        for(j=0;j<n;j++)

            Z[i][j] = X[i][j] - Y[i][j];

}

void multiply (int X[10][10], int Y[10][10], int Z[10][10], int m, int n)

{

    int i, j, k;

    for(i = 0; i < m; i++)

    {

        for(j = 0; j < n; j++)

        {

            Z[i][j] = 0;

            for(k = 0; k < n; k++)

            Z[i][j] += X[i][k] \* Y[k][j];

        }

    }

}

void transpose (int Y[10][10], int Z[10][10], int m, int n)

{

    int i, j;

    for(i = 0; i < m; i++)

        for(j = 0; j < n; j++)

            Z[i][j] = Y[j][i];

}

int main()

{

    int m, n, choice, A[10][10], B[10][10], C[10][10];

    printf("Enter the number of Rows :");

    scanf("%d", &m);

    printf("Enter the number of Columns :");

    scanf("%d", &n);

    printf("\nEnter Matrix A:\n");

    accept(A, m, n);

    printf("\n Enter Matrix B:\n");

    accept(B, m, n);

    do

    {

        printf("\nChoose the matrix operation,\n");

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

        printf("1. Addition\n");

        printf("2. Subtraction\n");

        printf("3. Multiplication\n");

        printf("4. Transpose\n");

        printf("5. Exit\n");

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

        printf("Enter your choice: ");

        scanf("%d", &choice);

        switch (choice)

        {

            case 1:

                add(A, B, C, m, n);

                printf("\nSum of matrix: \n");

                display (C, m, n);

                break;

            case 2:

                subtract(A, B, C, m, n);

                printf("\nSubtraction of matrix: \n");

                display (C, m, n);

                break;

            case 3:

                multiply (A, B, C, m, n);

                printf("\nMultiplication of matrix: \n");

                display (C, m, n);

                break;

            case 4:

                printf("\nTranspose of the first matrix: \n");

                transpose (A, C, m, n);

                display (C, m, n);

                printf("\nTranspose of the second matrix: \n");

                transpose (B, C, m, n);

                display (C, m, n);

                break;

            case 5:

                printf("Thank You.\n");

                exit (0);

            default:

                printf("Invalid input.\n");

                printf("Please enter the correct input.\n");

        }

    }while(1);

return 0;

}

**Experiment No 2 Code:**

*//Program Sparse matrix*

#include <stdio.h>

#include <stdlib.h>

void accept(int matrix[][100],int row,int col)

{

    for(int i=0; i<row; i++)

    {

        for(int j=0; j<col; j++)

        {

            printf("Enter the values [%d][%d]:",i ,j);

            scanf("%d", &matrix[i][j]);

        }

    }

}

void compact(int a[100][100], int b[100][100], int row, int col)

{

    b[0][0]=row;

    b[0][1]=col;

    int k=1;

    for(int i=0; i<row; i++)

    {

        for(int j=0; j<col; j++)

        {

            if (a[i][j] != 0)

            {

                b[k][0]=i;

                b[k][1]=j;

                b[k][2] = a[i][j];

                k++ ;

            }

        }

    }

    b[0][2] = k-1;

    int elm = b[0][2];

    if(elm <=0)

    {

        printf("Zero Matrix\n");

    }

    else

    {

        printf("Compact Matrix :\n");

        for(int i=0; i<elm+1; i++)

        {

            for(int j=0; j<3; j++)

            {

                printf("%d\t",b[i][j]);

            }

            printf("\n");

        }

    }

}

void simptrans(int a[100][100],int res[100][3])

{

    int row, col, t;

    row = a[0][0];

    col = a[0][1];

    t = a[0][2];

    res[0][0] = col;

    res[0][1] = row;

    res[0][2] = t;

    int q=1;

    for(int i=0; i<a[0][1]; i++)

    {

        for(int j=1; j<a[0][2]+1; j++)

        {

            if (a[j][1] == i)

            {

                res[q][0] = a[j][1];

                res[q][1] = a[j][0];

                res[q][2] = a[j][2];

                q += 1;

            }

        }

    }

    if(t<=0)

    {

        ;

    }

    else

    {

        printf("Simple Transpose :\n");

        for(int k=0; k<a[0][2]+1; k++)

        {

            for(int l=0; l<3; l++)

            {

                printf("%d\t",res[k][l]);

            }

            printf("\n");

        }

    }

}

void fasttrans(int a[100][100])

{

    int s[a[0][1]], t[a[0][1]+1];

    int row, col, elm;

    int r[100][3];

    row = a[0][0];

    col = a[0][1];

    elm = a[0][2];

    for(int i=0; i<a[0][1]; i++)

    {

        s[i] = 0;

    }

    for(int i=1; i<a[0][2]+1; i++)

    {

        s[a[i][1]] += 1;

    }

    t[0]=1;

    for(int i=1; i<=a[0][1]; i++)

    {

        t[i] = t[i-1] + s[i-1];

    }

    r[0][0] = col;

    r[0][1] = row;

    r[0][2] = elm;

    int j;

    for(int i=1; i<=a[0][2]; i++)

    {

        j = a[i][1];

        r[t[j]][0] = a[i][1];

        r[t[j]][1] = a[i][0];

        r[t[j]][2] = a[i][2];

        t[j]++;

    }

    if(a[0][2]<=0)

    {

        ;

    }

    else

    {

        printf("Fast Transpose :\n");

        for(int k=0; k<a[0][2]+1; k++)

        {

           for(int l=0; l<3; l++)

            {

                printf("%d\t",r[k][l]);

            }

            printf("\n");

        }

    }

}

void display(int matrix[100][100], int row, int col)

{

    printf("Your Entered Matrix is :\n");

    for(int i=0; i<row; i++)

    {

        for(int j=0; j<col; j++)

        {

            printf("%d\t",matrix[i][j]);

        }

        printf("\n");

    }

}

int main()

{

    int a [100][100], b[100][100], res[100][3], r1, c1, elm, choice,k,l;

    printf("Enter the number of Rows for the Matrix :");

    scanf("%d", &r1);

    printf("Enter the number of Columns for the Matrix :");

    scanf("%d", &c1);

    printf("Enter the Values for the Matrix :\n");

    accept(a, r1, c1);

    display(a, r1, c1);

    do

    {

        printf("Choose an option:\n");

        printf("==============================\n");

        printf("1. Convert to Compact\n");

        printf("2. Simple Transpose\n");

        printf("3. Fast Transpose\n");

        printf("4. Exit\n");

        printf("==============================\n");

        printf("Enter your choice:");

        scanf("%d", &choice);

        switch(choice)

        {

            case 1:

            compact(a,b, r1, c1);

            break;

            case 2:

            compact(a,b, r1, c1);

            printf("\n");

            simptrans(b, res);

            break;

            case 3:

            compact(a ,b, r1, c1);

            printf("\n");

            fasttrans(b);

            break;

            case 4:

            printf("Thank You!\n");

            exit(0);

            default:

            printf("Invalid input!\n");

            printf("Please enter the correct input.\n");

        }

    }while (1);

    return 0;

}

**Experiment No 3 Code:**

*//Program Searching and Sorting Algorithms*

#include<stdio.h>

struct student

{

  int rollno;

  char name[10];

  int marks;

};

void accept (struct student s[10], int n)

{

  int i;

  for (i = 0; i < n; i++)

    {

      scanf ("%d%s%d", &s[i].rollno, &s[i].name, &s[i].marks);

    }

}

void display (struct student s[10], int n)

{

  int i;

  printf ("Roll No\t  Name\tMarks\n");

  for (i = 0; i < n; i++)

    {

      printf (" %d\t  %s\t %d\n", s[i].rollno, s[i].name, s[i].marks);

    }

}

void linear (struct student s[10], int n, int key)

{

  int i,flag;

  printf ("\nEnter the roll number you want to search:\n");

  scanf ("%d", &key);

  for (i = 0; i < n; i++)

    {

      if (s[i].rollno == key)

    {

      printf ("Target key found!\n");

      printf ("Roll no\t  Name\tMarks\n");

      printf ("  %d \t   %s \t %d\n", s[i].rollno, s[i].name, s[i].marks);

      flag=1;

    }

    }

    if(flag!=1){

       printf ("Target key not found!\n");

    }

}

int binary (struct student s[10], int low, int high, int key, int n)

{

  int mid;

  low = 0;

  high = n - 1;

  printf ("Enter the roll number you want to search:\n");

  scanf ("%d", &key);

  while (low <= high)

    {

      mid = (low + high) / 2;

      if (s[mid].rollno == key)

    {

      return mid;

    }

      else if (key < s[mid].rollno){

       high = mid - 1;

      }

      else{

      low = mid + 1;

      }

    }

     return -1;

}

int recursive\_binary (struct student s[10], int low, int high, int key, int n)

{

  int mid;

  low = 0;

  high = n - 1;

  while (low <= high)

    {

      mid = (low + high) / 2;

      if (s[mid].rollno == key){

        return mid;

      }

      else if (key < s[mid].rollno){

        return recursive\_binary (s, mid + 1, high, key, n);

      }

      else{

        return recursive\_binary (s, low, mid - 1, key, n);

    }

  return -1;

}

}

void selectionsort(struct student s[10], int n){

    int i,j,minpos;

    struct student temp;

    for(i=0;i<(n-1);i++)

    {

        minpos=i;

        for(j=(i+1);j<n;j++)

        {

            if (s[j].rollno < s[minpos].rollno)

            {

             minpos=j;

            }

        }

          if(minpos!=i)

          {

              temp= s[i];

              s[i] = s[minpos];

              s[minpos] =temp;

          }

    }

     display(s,n);

}

void shellsort(struct student s[10], int n){

    int gap=n/2,swapped,i;

    struct student temp;

    do{

        do{

            swapped=0;

            for(i=0;i<n-gap;i++){

                if(s[i].rollno>s[i+gap].rollno){

                    temp= s[i];

                    s[i] = s[i+gap];

                    s[i+gap] =temp;

                    swapped++;

                }

            }

        }while(swapped==1);

    }while((gap=gap/2)>=1);

    display(s,n);

}

void insertionsort(struct student s[10], int n){

    struct student temp;

    int i,j;

    for(i=0;i<n;i++){

        temp= s[i];

        j = i-1;

        while(j>=0 && s[j].rollno > temp.rollno){

            s[j+1]=s[j];

            j=j-1;

        }

         s[j+1]=temp;

    }

     display(s,n);

}

int main ()

{

  int n, low, high, key, x, y, mid, choice, ch,c;

  struct student s[10];

  struct student temp[10];

  printf ("Enter the number of students: \n");

  scanf ("%d", &n);

  printf ("\nEnter the Roll No, Names and Marks respectively\n");

  accept (s, n);

  printf ("\nThe data you entered is:\n");

  display (s, n);

  do

    {

      printf ("\nEnter the operation you want to perform:\n");

      printf ("1.Searching\n2.Sorting\n");

      scanf ("%d", &ch);

      switch (ch)

    {

    case 1:

      do

        {

          printf ("\nDifferent searching operations -\n");

          printf

        ("1.Linear Search\n2.Binary Search(Non-recursive)\n3.Binary Search(Recursive)\n4.Exit");

          printf("\nEnter the type of search you want to execute :");

          scanf("%d", &choice);

          switch(choice)

        {

        case 1:

          linear (s, n, key);

          break;

        case 2:

          x = binary (s, low, high, key, n);

          if (x > 0)

            {

              printf ("Target key has been found!\n");

              printf ("Roll no\t  Name\tMarks\n");

              printf ("  %d \t  %s \t  %d\n", s[x].rollno, s[x].name,

                  s[x].marks);

            }

          else

            printf ("Target key has not been found!\n");

          break;

        case 3:

          printf ("\nEnter the roll number you want to search:\n");

          scanf ("%d", &key);

          y = recursive\_binary (s, low, high, key, n);

          if (y >= 0)

            {

              printf ("Target key has been found!\n");

              printf ("Roll no\t  Name\tMarks\n");

              printf ("  %d \t  %s \t  %d\n", s[y].rollno, s[y].name,

                  s[y].marks);

            }

          else

            printf ("Target key has not been found!\n");

          break;

        }

        }while (choice<4);

        break;

    case 2:  do

              {

             printf ("\nDifferent sorting operations -\n");

             printf("1.Selection Sorting\n2.Shell Sorting\n3.Insertion Sorting\n4.Exit\n");

             printf ("\nEnter the type of sorting you want to execute :");

             scanf ("%d", &c);

             switch (c)

             {

             case 1 : selectionsort(s,n);

                      break;

             case 2 : shellsort(s,n);

                      break;

             case 3 : insertionsort(s,n);

                      break;

             }

             }while(c<4);

    }

    }while (1);

      return 0;

}

**Experiment No 4 Code:**

*//Program Bucket Sort*

#include <stdio.h>

void accept (int n, int Marks[20])

{

  int i;

  for (i = 0; i < n; i++)

    {

      scanf ("%d", &Marks[i]);

    }

}

void display (int n, int Marks[20])

{

  int i;

  for (i = 0; i < n; i++)

    {

      printf ("%d\n", Marks[i]);

    }

}

void displayf (int n, int Marks[20])

{

  int i;

  for (i = n-1; i>(n-6); i--)

    {

      printf ("%d\n", Marks[i]);

    }

}

void bucketsort (int A[200], int n)

{

  int bucket[100], i =0, j = 0;

  for (i = 0; i < 100; i++)

    {

      bucket[i] = 0;

    }

  for (i = 0; i < n; i++)

    {

      ++bucket[A[i]];

    }

  for (i = 0, j = 0; i < 100; i++)

    {

      for (; bucket[i] > 0; --bucket[i])

{

  A[j] = i; j++;

}

}

}

  int main ()

  {

    int Marks[20], n;

    printf

      ("Enter the number of students whose percentage are to be entered:  ");

    scanf ("%d", &n);

    printf ("\nEnter the percentage of students:\n");

    accept (n, Marks);

    printf ("\nRecord before sorting :\n");

    display (n, Marks);

    bucketsort (Marks, n);

    printf ("\nRecord after sorting :\n");

    display (n, Marks);

    printf ("\nThe percentages of top 5 students are:\n");

    displayf (n, Marks);

    return 0;

  }