**DSA PROGRAMMING LAB-1,2,3**

**Before Link List**

• Name :HITU RAJ

• Roll no. :2005025

• Branch :CSE

**LAB 1**

/\*Q1.WAP in C to dynamically allocate memory using malloc or calloc to store  ‘N’ numbers entered by a user

and display all the numbers and the average of the numbers.\*/

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

int main()

{

    int n\_025, \*p\_025, sum\_025 = 0;

    float avg\_025;

    printf("how many elements are there");

    scanf("%d", &n\_025);

    p\_025 = (int \*)malloc(n\_025 \* sizeof(int));

    for (int i = 0; i < n\_025; i++)

    {

        printf("Enter the %d element ", i + 1);

        scanf(" %d", &\*(p\_025 + i));

        sum\_025 = sum\_025 + \*(p\_025+i);

    }

    avg\_025 = sum\_025 / (float)n\_025;

    printf("the elements are ");

    for (int i = 0; i < n\_025; i++)

    {

        printf(" %d ", \*(p\_025 + i));

        sum\_025 = sum\_025 + \*p\_025;

    }

    printf("\nthe avg of the entered data is %f", avg\_025);

    return 0;

}

**OUTPUT-1**



//Q2.Find the largest element in an array using Dynamic Memory Allocation.

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

int main()

{

    int n\_025, \*p\_025, max\_025;

    printf("how many elements are there");

    scanf("%d", &n\_025);

    p\_025 = (int \*)malloc(n\_025 \* sizeof(int));

    for (int i = 0; i < n\_025; i++)

    {

        printf("Enter the %d element ", i + 1);

        scanf(" %d", &\*(p\_025 + i));

    }

    max\_025 = \*p\_025;

    for (int i = 0; i < n\_025; i++)

    {

        if (max\_025 <= \*(p\_025 + i))

            max\_025 = \*(p\_025 + i);

    }

    printf("the largest no. is %d", max\_025);

    return 0;

}

OUTPUT-2



/\*Q3.WAP to read and print the details of 5 CS student details

using Dynamic Memory Allocation (roll, age, cgpa, subject).

\*/

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

struct stud

{

    int roll\_025, age\_025;

    int cgpa\_025;

    char sub\_025[10];

};

int main()

{

    int n;

    struct stud \*s[10];

    printf("how many students are there");

    scanf("%d", &n);

    for (int i = 0; i < n; i++)

    {

        s[i] = (struct stud \* )malloc(n \* sizeof(struct stud));

    }

    printf("Enter the details ");

    for (int i = 0; i < n; i++)

    {

        printf("Enter the roll ,age, sub  cgpa, ");

        scanf("%d%d%s%d",&s[i]->roll\_025,&s[i]->age\_025, &s[i]->sub\_025, &s[i]->cgpa\_025);

    }

    for (int i = 0; i < n; i++)

    {

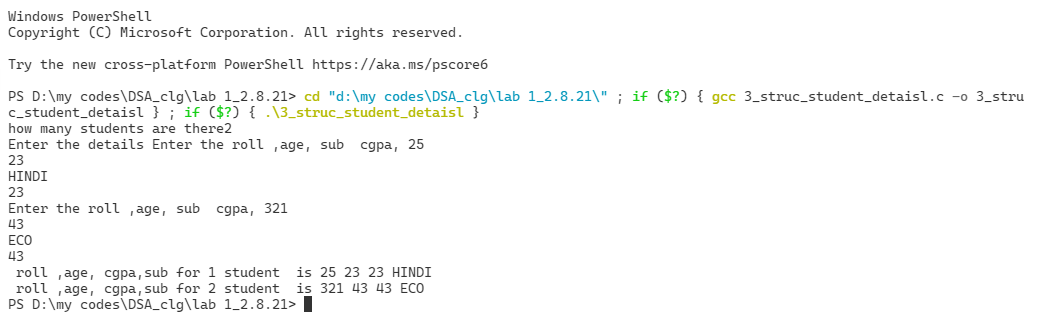
        printf(" roll ,age, cgpa,sub for %d student  is %d %d %d %s \n", i + 1, s[i]->roll\_025, s[i]->age\_025, s[i]->cgpa\_025, s[i]->sub\_025);

    }

    return 0;

}

OUTPUT-3



//Q4.WAP to find the standard deviation of 20 numbers.

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

int main()

{

    int n\_025, \*p\_025, sum\_025 = 0, mean\_025;

    double sqr\_025 = 0.0, sd\_025 = 0.0;

    printf("how many no. u have");

    scanf("%d", &n\_025);

    p\_025 = (int \*)malloc(sizeof(int) \* n\_025);

    printf("enter the no.s ");

    for (int i = 0; i < n\_025; i++)

    {

        scanf("%d", &p\_025[i]);

        sum\_025 = sum\_025 + p\_025[i];

    }

    mean\_025 = sum\_025 / n\_025;

    for (int i = 0; i < n\_025; i++)

    {

        double t = p\_025[i] - mean\_025;

        sqr\_025 = pow(t, 2);

        sd\_025 = sd\_025 + sqr\_025;

    }

    sd\_025 = sd\_025 / (n\_025 - 1);

    printf("the standard deviation is %lf ", sqrt(sd\_025));

    return 0;

}

OUTPUT-4



//WAP to arrange the elements of a dynamic

//array such that even numbers are followed by odd numbers.

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

int main()

{

    int n\_025, \*p\_025 ;

    printf("how many elements are there");

    scanf("%d", &n\_025);

    p\_025 = (int \*)malloc(n\_025 \* sizeof(int));

    for (int i = 0; i < n\_025; i++)

    {

        printf("Enter the %d element ", i + 1);

        scanf(" %d", &p\_025[i]);

    }

    for (int i = 0; i < n\_025 / 2; i++)

    {

        if (p\_025[i] % 2 == 0 && p\_025[n\_025 - 1 - i] % 2 != 0)

        {

        }

        else if (p\_025[i] % 2 != 0 && p\_025[n\_025 - 1 - i] % 2 == 0)

        {

            int temp = \*(p\_025 + i);

            \*(p\_025 + i) = \*(p\_025 + (n\_025 - i - 1));

            \*(p\_025 + (n\_025 - i - 1)) = temp;

        }

        else if (p\_025[i] % 2 == 0 && p\_025[n\_025 - 1 - i] % 2 == 0)

        {

            int j = i + 1, c = 0;

            while (c ==0&&j<=n\_025/2)

            {

                if (p\_025[j] % 2 != 0)

                {

                    int temp = \*(p\_025 + j);

                    \*(p\_025 + j) = \*(p\_025 + (n\_025 - i - 1));

                    \*(p\_025 + (n\_025 - i - 1)) = temp;

                    c = 1;

                }

                j++;

            }

        }

        else if (p\_025[i] % 2 != 0 && p\_025[n\_025 - 1 - i] % 2 != 0)

        {

            int j = n\_025 - 2 - i, c = 0;

            while (c == 0&&j>=n\_025/2)

            {

                if (p\_025[j] % 2 == 0)

                {

                    int temp = \*(p\_025 + j);

                    \*(p\_025 + j) = \*(p\_025 + i);

                    \*(p\_025 + i) = temp;

                    c = 1;

                }

                j--;

            }

        }

    }

    printf("even followed by odd is ");

    for (int i = 0; i < n\_025; i++)

    {

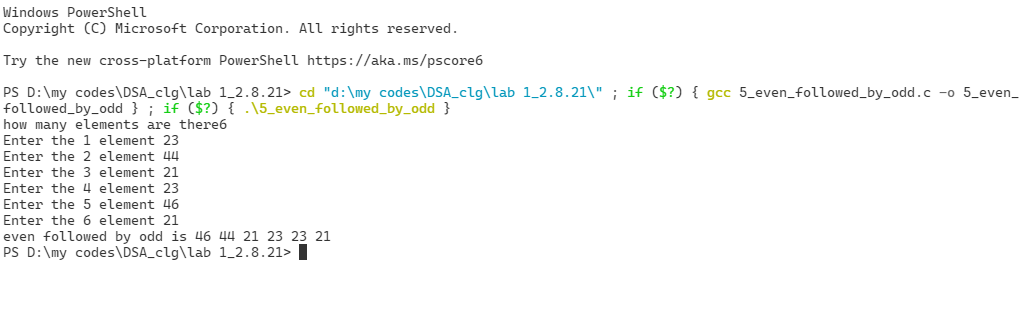
        printf("%d ", \*(p\_025 + i));

    }

    return 0;

}

OUTPUT-5



//Q6.WAP to reverse the contents of a dynamic array of ‘N’ elements.

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

int main()

{

    int n\_025, \*p\_025;

    printf("how many elements are there");

    scanf("%d", &n\_025);

    p\_025 = (int \*)malloc(n\_025 \* sizeof(int));

    for (int i = 0; i < n\_025; i++)

    {

        printf("Enter the %d element ", i + 1);

        scanf(" %d", &\*(p\_025 + i));

    }

    for (int i = 0; i < n\_025/2; i++)

    {

       int temp=\*(p\_025+i);

       \*(p\_025+i)=\*(p\_025+(n\_025-i-1));

       \*(p\_025+(n\_025-i-1))=temp;

    }

    printf("the reverse array is ");

    for (int i = 0; i < n\_025; i++)

    {

        printf("%d ", \*(p\_025+i) );

    }

    return 0;

}

OUTPUT-6



**LAB-2**

/\*Q1. Write a program using C to allocate memory dynamically for 2D array ,read the values and display them using 3 functions separately

I. Keeping no columns fixed and using variable no of rows

II. Keeping no rows fixed and using variable no of columns

III. Using both variable no of rows and columns(use double pointer)

\*/

#include <stdio.h>

#include <stdlib.h>

void rfcv()

{

    printf(".....THIS IS  2D DMA WITH fix Row & Var Column.....\n");

    int r\_025, c\_025;

    printf("how many columns you have ");

    scanf("%d", &c\_025);

    int \*p\_025[10], l = 1;

    for (int j = 0; j < 5; j++)

    {

        p\_025[j] = (int \*)malloc(c\_025 \* sizeof(int));

    }

    for (int i = 0; i < 5; i++)

    {

        for (int j = 0; j < c\_025; j++)

        {

            p\_025[i][j] = l;

            l++;

        }

    }

    for (int i = 0; i < 5; i++)

    {

        for (int j = 0; j < c\_025; j++)

        {

            printf("%d ", p\_025[i][j]);

        }

        printf("\n");

    }

}

void rvcf()

{

    printf(".....THIS IS  2D DMA WITH var Row & fix Column.....\n");

     int(\*p\_025)[5];

    int r\_025, l = 1;

    printf("how many rows you have ");

    scanf("%d", &r\_025);

    p\_025 = (int(\*)[5])malloc(r\_025 \* 5 \* sizeof(int));

    for (int i = 0; i < r\_025; i++)

    {

        for (int j = 0; j < 5; j++)

        {

            p\_025[i][j] = l;

            l++;

        }

    }

    for (int i = 0; i < r\_025; i++)

    {

        for (int j = 0; j < 5; j++)

        {

            printf("%d ", p\_025[i][j]);

        }

        printf("\n");

    }

}

void rvcv()

{

    printf(".....THIS IS  2D DMA WITH var Row & Var Column.....\n");

    int r\_025, c\_025, l = 1;

    printf("how many rows you have ");

    scanf("%d", &r\_025);

    printf("how many columns you have ");

    scanf("%d", &c\_025);

    int \*\*p\_025;

    p\_025 = (int \*\*)malloc(r\_025 \* sizeof(int));

    for (int j = 0; j < r\_025; j++)

    {

        p\_025[j] = (int \*)malloc((j + 1) \* sizeof(int));

    }

    for (int i = 0; i < r\_025; i++)

    {

        for (int j = 0; j < c\_025; j++)

        {

            p\_025[i][j] = l;

            l++;

        }

    }

    for (int i = 0; i < r\_025; i++)

    {

        for (int j = 0; j < c\_025; j++)

        {

            printf("%d ", p\_025[i][j]);

        }

        printf("\n");

    }

}

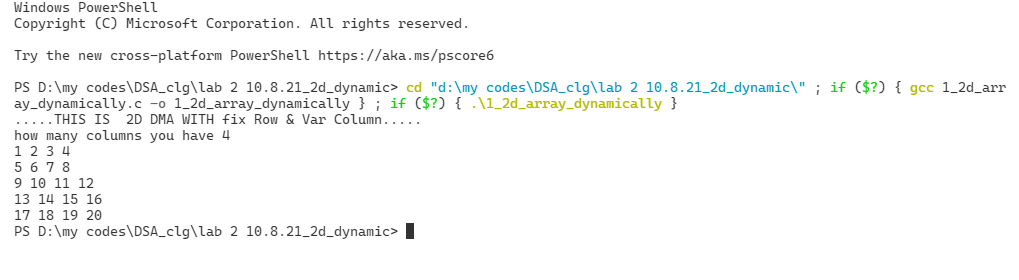
int main()

{

    rfcv();

    return 0;

} OUTPUT-1



/\*Q2 Let A be nXn square dynamic matrix. WAP by using appropriate user defined functions for the following:

a. Find the number of nonzero elements in A

B. Find the sum of the elements above the leading diagonal.

C. Display the elements below the minor diagonal.

D. Find the product of the diagonal elements.

\*/

#include <stdio.h>

#include <stdlib.h>

void zero\_el(int \*a[10], int c\_025)

{

    int count = 0;

    for (int i = 0; i < c\_025; i++)

    {

        for (int j = 0; j < c\_025; j++)

        {

            if (a[i][j] == 0)

            {

                count++;

            }

        }

    }

    printf("no. of zero element is %d \n\n", count);

}

void sumabovediad(int \*a[10], int c\_025)

{

    int sum = 0;

    for (int i = 0; i < c\_025; i++)

    {

        for (int j = 0; j < c\_025; j++)

        {

            if (j > i)

            {

                sum = sum + a[i][j];

            }

        }

    }

    printf("sum of elements above leading diagnol is %d\n\n", sum);

}

void disp\_b\_lead\_diag(int \*a[10], int c\_025)

{

    printf("the  elements below leading disgnol are :");

    for (int i = 0; i < c\_025; i++)

    {

        for (int j = 0; j < c\_025; j++)

        {

            if (i > j)

            {

                printf("%d ", a[i][j]);

            }

        }

    }

    printf("\n\n");

}

void diag\_prod(int \*a[10], int c\_025)

{

    int prod = 1;

    for (int i = 0; i < c\_025; i++)

    {

        for (int j = 0; j < c\_025; j++)

        {

            if (i ==j)

            {

                prod = prod \* a[i][j];

            }

            else if ((i + j) == (c\_025 - 1))

            {

                prod = prod \* a[i][j];

            }

        }

    }

    printf("the prod of the diag elements are %d \n\n:", prod);

}

int main()

{

    int \*p\_025[10], c\_025, l = 0;

    printf("how many columns you have ");

    scanf("%d",&c\_025);

    for (int i = 0; i < c\_025; i++)

    {

        p\_025[i] = (int \*)malloc(c\_025 \* sizeof(int));

    }

    printf("Enter elements \n");

    for (int i = 0; i < c\_025; i++)

    {

        for (int j = 0; j < c\_025; j++)

        {

            scanf("%d",&p\_025[i][j]);

        }

    }

    for (int i = 0; i < c\_025; i++)

    {

        for (int j = 0; j < c\_025; j++)

        {

            printf("%d ", p\_025[i][j]);

        }

        printf("\n");

    }

zero\_el(p\_025,c\_025);

sumabovediad(p\_025,c\_025);

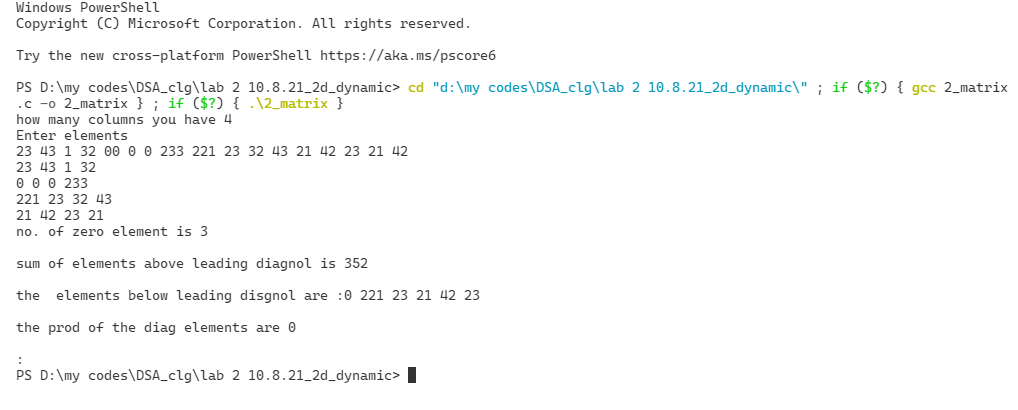
disp\_b\_lead\_diag(p\_025,c\_025);

diag\_prod(p\_025,c\_025);

    return 0;

}

OUTPUT-2



/\*Q3. WAP to swap all the elements in the 1st column with all the corresponding elements in the last column, and 2nd column with the second last column\

 and 3rd with 3rd last etc. of a 2-D dynamic array.  Display the matrix.

\*/

#include <stdio.h>

#include <stdlib.h>

void swap(int \*a[10], int r\_025, int c\_025)

{

    for (int i = 0; i < c\_025/2; i++)

    {

        for (int j = 0; j < r\_025; j++)

        {

            int temp = a[j][i];

            a[j][i] = a[j][c\_025 - 1-i];

            a[j][c\_025 - 1-i] = temp;

        }

    }

    for (int i = 0; i < r\_025; i++)

    {

        for (int j = 0; j < c\_025; j++)

        {

            printf("%d ", a[i][j]);

        }

        printf("\n");

    }

}

int main()

{

    int r\_025, c\_025, l = 1;

    printf("how many rows you have ");

    scanf("%d", &r\_025);

    printf("how many columns you have ");

    scanf("%d", &c\_025);

    int \*\*p\_025;

    p\_025 = (int \*\*)malloc(r\_025 \* sizeof(int));

    for (int j = 0; j < r\_025; j++)

    {

        p\_025[j] = (int \*)malloc(c\_025 \* sizeof(int));

    }

    for (int i = 0; i < r\_025; i++)

    {

        for (int j = 0; j < c\_025; j++)

        {

            p\_025[i][j] = l;

            l++;

        }

    }

    for (int i = 0; i < r\_025; i++)

    {

        for (int j = 0; j < c\_025; j++)

        {

            printf("%d ", p\_025[i][j]);

        }

        printf("\n");

    }

    printf("\n");

swap(p\_025,r\_025,c\_025);

return 0;

}

OUTPUT-3



**LAB-3**

//Q1.   WAP to add two distances (in kilometre-meter) by

//passing structure to a function.

#include <stdio.h>

struct dist

{

    int kilometre;

    int metre;

} ;

void add\_dis(struct dist s[2])

{

    int kilometre, metre;

    metre = s[0].metre + s[1].metre;

    kilometre = s[0].kilometre + s[1].kilometre + metre / 1000;

    metre = metre % 1000;

    printf("the added value of distance are %d kilometre %d metre ", kilometre, metre);

}

int main()

{ struct dist s[2];

    for (int i = 0; i < 2; i++)

    {

        printf("enter the distance %d in kilometre and metre ", i + 1);

        scanf("%d %d", &s[i].kilometre, &s[i].metre);

    }

add\_dis(s);

    return 0;

}

OUTPUT-1



//  Q2• You are given an array of 0s and 1s in random order.

// Segregate 0s on left side and 1s on right side of the array.

// Traverse array only once.

#include <stdio.h>

#include <stdlib.h>

int main()

{

    int \*p, n;

    printf("how many elements you have in the array ");

    scanf("%d", &n);

    p = (int \*)malloc(n \* sizeof(int));

    printf("Enter the elements ");

    for (int i = 0; i < n; i++)

    {

        scanf("%d",&p[i]);

    }

    for (int i = 0; i < n; i++)

    {

        printf("%d ", p[i]);

    }

    printf("\n");

    for (int i = 0, j = n-1; i <= n / 2 && j >= n / 2;)

    {

        if (p[i] == 1 && p[j] == 0)

        {

            int temp = p[i];

            p[i] = p[j];

            p[j] = temp;

            i++;

            j--;

        }

        else if (p[i] == 0 && p[j] == 0)

        {

            int temp = p[i + 1];

            p[i + 1] = p[j];

            p[j] = temp;

            i++;

        }

        else if (p[i] == 1 && p[j] == 1)

        {

            int temp = p[i];

            p[i] = p[j-1];

            p[j - 1] = temp;

            j--;

        }

        else

        {

            i++;

            j--;

        }

    }

    for (int i = 0; i < n; i++)

    {

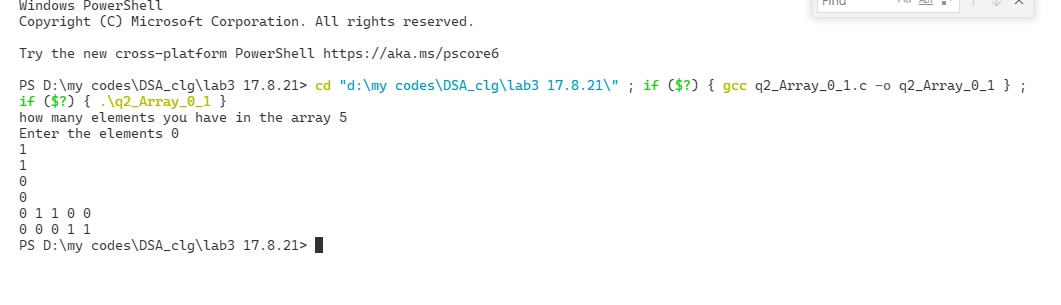
        printf("%d ", \*(p + i));

    }

return 0;

}

OUTPUT-2



//Q3•   Given an unsorted dynamic array arr and two numbers x and y, find the minimum distance between x and y

//in arr. The array might also contain duplicates. You may assume that both x and y are different and present in arr.

//Input: arr[] = {3, 5, 4, 2, 6, 5, 6, 6, 5, 4, 8, 3}, x = 3, y = 6

//Output: Minimum distance between 3 and 6 is 4.

#include <stdio.h>

#include <stdlib.h>

int main()

{

    int \*p, n, x, y,uplimit,lowerlimit;

    printf("how many elements you have in the array ");

    scanf("%d", &n);

    p = (int \*)malloc(n \* sizeof(int));

    printf("Enter the elements ");

    for (int i = 0; i < n; i++)

    {

        scanf("%d", &p[i]);

    }

    for (int i = 0; i < n; i++)

    {

        printf("%d ", p[i]);

    }

    printf("\nenter 2 elements you need to find the distance ");

    scanf("%d %d", &x, &y);

    printf("\n");

    for (int i = 0; i <= n; i++)

    {

        if (p[i] == x)

        {

            lowerlimit = i;

            break;

        }

    }

    for (int i = 0; i < n; i++)

    {

      if (p[i] == y)

        {

            uplimit = i;

            break;

        }

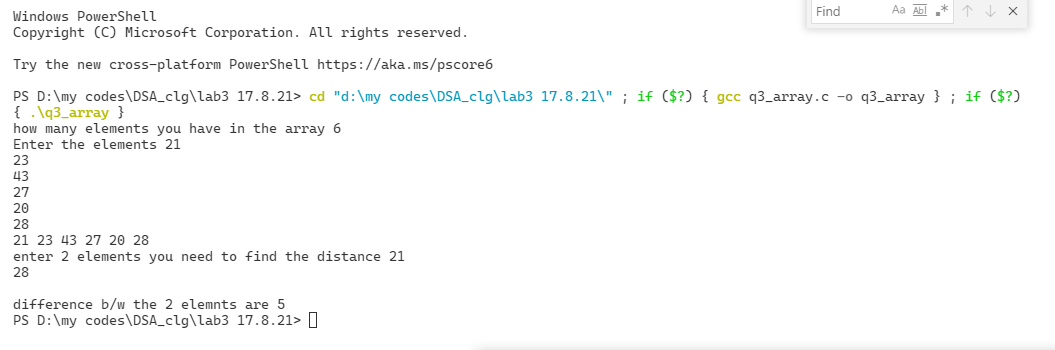
    }

    printf("difference b/w the 2 elemnts are %d ",uplimit-lowerlimit);

    return 0;

}

OUTPUT-3



//Q4•   Write a program in C to find the LCM ,HCF,GCD of two numbers using recursion

//by using menu driven switch case and functions.

#include <stdio.h>

void hcf(int a, int b, int i)

{     static int thcf = 1;

    if (a % i == 0 && b % i == 0)

        {

            thcf = thcf \* i;

            hcf(a / i, b / i, 2);

        }

        else if (i <= a && i <= b)

            hcf(a,b,++i);

        else

        {

            printf("\nhcf of 2 no. is %d", thcf);

        }

}

  void lcm(int a, int b, int i)

{     static int tlcm = 1;

    if (a % i == 0 && b % i == 0)

        {

            tlcm = tlcm \* i;

            lcm(a / i, b / i, 2);

        }

        else if (i <= a && i <= b)

            lcm(a,b,++i);

        else

        {

            printf("\n lcm of 2 no. is %d", tlcm\*a\*b);

        }

}

int main()

{

    int a, b;

    printf("Enter 2 no.s you want to calculate hcf anf lcm");

    scanf("%d%d", &a, &b);

    lcm(a, b, 1);

    hcf(a, b, 1);

    return 0;

}

OUTPUT-4

