/\*

Program : 1

Author  : Anish

Topic   : Write a C program using OpenMP features to create two parallel threads.

            The first thread should push the first ‘N’ natural numbers into a stack

            in sequence, and the second thread should pop the numbers from the stack.

\*/

#include<stdio.h>

#include<omp.h>

#include<stdlib.h>

int main()

{

    int n,a;

    printf("\n ENTER THE VALUE OF N \n");

    scanf("%d",&n);

    int id,d,Q[n],top=-1;

    omp\_set\_dynamic(0);

    #pragma omp parallel num\_threads(2)

    {

        id=omp\_get\_thread\_num();

        if(id==0) //push

        {

            while(1)

            {

                #pragma omp critical

                {

                    if(top<n-1)

                    {

                        printf("\n ENTER A NUMBER \n");

                        scanf("%d",&a);

                        Q[++top]=a;

                        printf("\n INSERTED ITEM IS %d",a);

                    }

                    else

                        printf("\n  NO SPACE");

                    fgetc(stdin);

                }

            }

        }

        else

        {

            while(1) //pop

            {

                #pragma omp critical

                {

                    if(top!=-1)

                    {

                        d=Q[top];

                        top--;

                        printf("\n DELETED ITEM IS %d",d);

                    }

                    else

                        printf("\n  NO ITEMS TO DELETE");

                    fgetc(stdin);

                }

            }

        }

    }

    return 0;

}

/\*

Program : 2

Author  : Gyan

Topic   : Write a C program using OpenMP features to create three parallel threads.

            The first thread should display the value of a global variable, ‘X’; the

            second thread should increment the value of the same global variable, ‘X’

            and the third thread should decrement the value of ‘X’.

\*/

#include<stdio.h>

#include<omp.h>

int main()

{

    int a=0,id;

    omp\_set\_dynamic(0);

    #pragma omp parallel num\_threads(3)

    {

        id=omp\_get\_thread\_num();

        if(id == 0)

        {

            while(1)

            {

                #pragma omp critical

                    printf("Data =  %d\n",a);

            }

        }

        else if(id == 1)

        {

            while(1)

            {

                #pragma omp critical

                {

                    a++;

                    printf("Increment \n");

                }

            }

        }

        else

        {

            while(1)

            {

                #pragma omp critical

                {

                    a--;

                    printf("Decrement \n");

                }

            }

        }

    }

    return 0;

}

/\*

Program : 3

Author  : Anish

Topic   : Write a C program using OpenMP features to create two parallel threads.

            The first thread should insert the first ‘N’ natural numbers into a queue in sequence,

            and the second thread should remove the numbers from the queue.

\*/

#include<stdio.h>

#include<omp.h>

#include<stdlib.h>

int main()

{

    int n;

    printf("\n ENTER THE VALUE OF N \n");

    scanf("%d",&n);

    int id,d,Q[n],rear=-1,front=0,i=1;

    omp\_set\_dynamic(0);

    #pragma omp parallel num\_threads(2)

    {

        id=omp\_get\_thread\_num();

        if(id==0) //insert

        {

            while(1)

            {

                #pragma omp critical

                {

                    if(rear<n-1)

                    {

                        Q[++rear]=i;

                        printf("\n INSERTED ITEM IS %d",i);

                        i++;

                    }

                    else

                        printf("\n  NO SPACE");

                    fgetc(stdin);

                }

            }

        }

        else

        while(1) //pop

        {

            #pragma omp critical

            {

                if(front<=rear)

                {

                    d=Q[front];

                    front++;

                    printf("\n DELETED ITEM IS %d",d);

                }

                else

                    printf("\n  NO ITEMS TO DELETE");

                fgetc(stdin);

            }

        }

    }

    return 0;

}

/\*

Program : 4

Author  : Gyan

Topic   : Write a C program using OpenMP features to find the sum of

            two matrices in linear time. The program should then find row wise

            average of the sum matrix.

\*/

#include<stdio.h>

#include<omp.h>

int main()

{

    int n, m, i, j, sum;

    omp\_set\_dynamic(0);

    m = omp\_get\_num\_threads();

    omp\_set\_num\_threads(m);

    printf("enter the dimension of the matrix:");

    scanf("%d", &n);

    int a[n][n], b[n][n], c[n][n], avg[n];

    printf("enter matrix a :: \n");

    for(i = 0 ; i < n ; i++)

    {

        for(j = 0 ; j < n ; j++)

            scanf("%d", &a[i][j]);

    }

    printf("enter matrix b :: \n");

    for(i = 0 ; i < n ; i++)

    {

        for(j = 0 ; j < n ; j++)

            scanf("%d", &b[i][j]);

    }

    for(i = 0 ; i < n ; i++)

    {

        #pragma parallel for shared(a, b, c) private(j)

        for(j = 0 ; j < n ; j++)

            c[i][j] = a[i][j] + b[i][j];

    }

    for(i = 0 ; i < n ; i++)

    {

        sum = 0;

        #pragma parallel for shared(c, sum) private(j)

        for(j = 0 ; j < n ; j++)

            sum += c[i][j];

        avg[i] = sum / n;

    }

    printf("sum matrix ::  \n");

    for(i = 0 ; i < n ; i++)

    {

        for(j = 0 ; j < n ; j++)

            printf("%d \t", c[i][j]);

        printf("\n");

    }

    printf("row wise avg ::  ");

    for(j = 0 ; j < n ; j++)

        printf("%d \t", avg[j]);

    return 0;

}

/\*

Program : 5

Author  : Anish

Topic   : Write a C program using OpenMP features to create two parallel threads

            to simulate a linear queue. The first thread should implement the

            insert operation on the linear queue. The second thread should

            implement the remove operation on the linear queue. Both the threads

            should run infinitely.

\*/

#include<stdio.h>

#include<omp.h>

#include<stdlib.h>

int main()

{

    int n,a,num = 0;

    printf("\n ENTER THE VALUE OF N \n");

    scanf("%d",&n);

    int id,d,Q[n],rear=-1,front=-1;

    omp\_set\_dynamic(0);

    #pragma omp parallel num\_threads(2)

    {

        id=omp\_get\_thread\_num();

        if(id==0) //insert

        {

            while(1)

            {

                #pragma omp critical

                {

                    if(rear<n-1)

                    {

                        Q[++rear]=num;

                        printf("\n INSERTED ITEM IS %d",num);

                        num++;

                    }

                    else

                        printf("\n  NO SPACE");

                    //fgetc(stdin);

                }

            }

        }

        else

        {

            while(1) //pop

            {

                #pragma omp critical

                {

                    if(front == rear && front != -1)

                    {

                        d=Q[front];

                        front = -1;

                        rear = -1;

                        printf("\n DELETED ITEM IS %d",d);

                    }

                    if(front<rear)

                    {

                        d=Q[front];

                        front++;

                        printf("\n DELETED ITEM IS %d",d);

                    }

                    else

                        printf("\n  NO ITEMS TO DELETE");

                    //fgetc(stdin);

                }

            }

        }

    }

    return 0;

}

/\*

Program : 6

Author  : Anish

Topic   : Write a C program using OpenMP features to implement one reader

        and one writer threads. The reader thread should display the value

        of a global variable, whereas the writer thread should increment the

        value of the global variable. Both the threads should run infinitely.

\*/

#include<stdio.h>

#include<omp.h>

int main()

{

    int a=10,id;

    omp\_set\_dynamic(0);

    #pragma omp parallel num\_threads(2)

    {

        id=omp\_get\_thread\_num();

        if(id==0) //reader

        {

            while(1)

            {

                #pragma omp critical

                {

                    printf("\n READER THREAD %d",a);

                }

            }

        }

        else

        {

            while(1) //writer

            {

                #pragma omp critical

                {

                    ++a;

                    printf("\n WRITER THREAD");

                }

            }

        }

    }

    return 0;

}

/\*

Program : 7

Author  : Debottam

Topic   : Write a C program using OpenMP features to find the product of two nxn matrices.

            The program should then find the sum of all the elements of the product matrix.

\*/

#include <stdio.h>

#include <omp.h>

#define N 3

int A[3][3];

int B[3][3];

int C[3][3];

int main()

{

   int i,j,k,l=1,s=0;

   int m= omp\_get\_num\_procs();

   omp\_set\_num\_threads(m);

   // Generating sample dataset

   for (i= 0; i< N; i++)

   {

        for (j= 0; j< N; j++)

        {

            A[i][j] = l;

            l++;

            B[i][j] = l;

            l++;

        }

   }

   printf("Matrix A: \n");

   for (i= 0; i< N; i++)

   {

        for (j= 0; j< N; j++)

            printf("%d\t",A[i][j]);

        printf("\n");

   }

   printf("Matrix B: \n");

   for (i= 0; i< N; i++)

   {

        for (j= 0; j< N; j++)

            printf("%d\t",B[i][j]);

        printf("\n");

   }

   #pragma omp parallel for shared(A,B,C) private(i,j,k)

   for (i = 0; i < N; ++i)

   {

        for (j = 0; j < N; ++j)

        {

            for (k = 0; k < N; ++k)

                C[i][j] += A[i][k] \* B[k][j];

        }

   }

   printf("Product matrix C: \n");

   for (i= 0; i< N; i++)

   {

        for (j= 0; j< N; j++)

        {

            printf("%d\t",C[i][j]);

            s=s+C[i][j];

        }

        printf("\n");

    }

    printf("Sum of all elements of the product matrix = %d",s);

    return 0;

}

/\*

Program : 8

Author  : Debottam

Topic   : Write a C program using OpenMP features to find the cross

            product of two vectors in constant time complexity.

\*/

#include <stdio.h>

#include <omp.h>

int main()

{

    int A[]={3,-5,4},i;

    int B[]={2,6,5},C[3],D=0;

    int m= omp\_get\_num\_procs();

    omp\_set\_num\_threads(m);

    #pragma omp parallel for shared(C) private(i)

    for(i=0;i<3;i++)

    {

        C[i]=A[(i+1)%3]\*B[(i+2)%3]-A[(i+2)%3]\*B[(i+1)%3];

    }

    printf("Cross product, C = ");

    for (i= 0; i< 3; i++)

        printf("%d\t",C[i]);

     printf("\n");

     return 0;

 }

/\*

Program : 9

Author  : Soumili

Topic   : Write a C program using OpenMP features to find the row wise/column

            wise sum of a matrix in linear time complexity.

\*/

#include<stdio.h>

#include<omp.h>

int main()

{

    int i,j,k,m,sum,sum1;

    int A[3][3]={1,2,3,

                 4,5,6,

                 7,8,9};

    omp\_set\_dynamic(0);

    m=omp\_get\_num\_procs();

    omp\_set\_num\_threads(m);

    #pragma omp parallel for shared(A) private(i,j, sum, sum1)

    for(i=0;i<3;i++)

    {

        sum =0,sum1=0;

        for(j=0;j<3;j++)

        {

            sum=sum+A[j][i] ;

            sum1=sum1+A[i][j];

        }

        printf(" sum of %d column is %d from thread %d of %d\n",i+1,sum,omp\_get\_thread\_num(),omp\_get\_num\_threads());

        printf(" sum of %d row is %d from thread %d of %d\n",i+1,sum1,omp\_get\_thread\_num(),omp\_get\_num\_threads());

    }

    return 0;

}

/\*

Program : 10

Author  : Gyan

Topic   : 10.Write a C program using OpenMP features to find the sum

            of the first ‘n’ terms of the following series:

            X.(X+1) + (X+1).(X+2) + (X+2)(X+3) + …

            Where, ‘X’ and ‘n’ are two values input by the user.

\*/

#include<stdio.h>

#include<omp.h>

int main()

{

    int x, n, sum, i, m;

    omp\_set\_dynamic(0);

    m = omp\_get\_num\_threads();

    omp\_set\_num\_threads(m);

    printf("enter value of x and n:: ");

    scanf("%d%d", &x, &n);

    sum = 0;

    #pragma parallel for reduction(+:sum)

    for(i = 0 ; i < n ; i++)

        sum += (x+i)\*(x+i+1);

    printf("The sum of the series is :: %d \n", sum);

    return 0;

}

/\*

Program : 11

Author  : Gyan

Topic   : Write a C program using OpenMP features to find the determinant of a 3x3 matrix.

\*/

#include <stdio.h>

#include <omp.h>

int main()

{

    int i, m, D=0;

    int a[3][3] = { {1, 2, 3},

                    {4, 5, 6},

                    {7, 8, 10}};

    m = omp\_get\_num\_procs();

    omp\_set\_num\_threads(m);

    #pragma omp parallel for shared(a, D) private(i)

    for(i=0;i<3;i++)

        D += a[0][i] \*( a[1][(i+1)%3] \* a[2][(i+2)%3] - a[1][ (i+2)%3] \* a[2][(i+1)%3 ]);

    printf("\nDeterminant = %d\n",D);

    return 0;

}

/\*

Program : 12

Author  : Soumili

Topic   : Write a C program using OpenMP features to

            find the column wise average of a matrix in linear time complexity.

\*/

#include<stdio.h>

#include<omp.h>

int main()

{

    int i,j,k,m,sum,avgc;

    int A[3][3]={1,2,3,

           4,5,6,

           7,8,9};

    omp\_set\_dynamic(0);

    m=omp\_get\_num\_procs();

    omp\_set\_num\_threads(m);

    #pragma omp parallel for shared(A) private(i,j,sum,avgc)

    for(i=0;i<3;i++)

    {

        sum =0;

        for(j=0;j<3;j++)

        {

            sum=sum+A[j][i];

        }

        avgc=sum/3;

        printf(" Average of %d column is %d from thread %d of %d\n",i+1,avgc,omp\_get\_thread\_num(),omp\_get\_num\_threads());

    }

    return 0;

}

/\*

Program : 13

Author  : Debottam

Topic   : Write a C program using Open MP features to find the dot

            product of two vectors of size n each in constant time complexity.

            [Hint: Dot product = Ʃ(A[i]\*B[i])]

\*/

#include <stdio.h>

#include <omp.h>

#define N 3

int main()

{

    int A[]={3,-5,4},i;

    int B[]={2,6,5},C[N],D=0;

    int m= omp\_get\_num\_procs();

    omp\_set\_num\_threads(m);

    #pragma omp parallel for shared(D) private(i)

    for(i=0;i<N;i++)

    {

        D=D+(A[i]\*B[i]);

    }

    printf("\nDot product, D = A.B = %d\n",D);

    return 0;

}

/\*

Program : 14

Author  : Debottam

Topic   : Write a C program using OpenMP features to find the cross product

            of two vectors of size n each in constant time complexity.

            [Hint: Cross product C[i] = (A[i]\*B[i])]

\*/

#include <stdio.h>

#include <omp.h>

#define N 3

int main()

{

    int A[]={3,-5,4},i;

    int B[]={2,6,5},C[N],D=0;

    int m= omp\_get\_num\_procs();

    omp\_set\_num\_threads(m);

    #pragma omp parallel for shared(C) private(i)

    for(i=0;i<N;i++)

    {

        C[i]=A[(i+1)%N]\*B[(i+2)%N]-A[(i+2)%N]\*B[(i+1)%N];

    }

    printf("Cross product, C = ");

    for (i= 0; i< N; i++)

        printf("%d\t",C[i]);

    printf("\n");

    return 0;

 }