**PP LAB WEEK-2**

# DSE VI-A2 Divansh Prasad 210968140

1) Write a program in C to reverse the digits of the following integer array of size 9. Initialise the input array to the following values.

Input array: 18, 523, 301, 1234, 2, 14, 108, 150, 1928

Output array: 81, 325, 103, 4321, 2, 41, 801, 51, 8291

#include <stdio.h>

#include <omp.h>

#include <time.h>

#include <windows.h>

int main(){

clock\_t start, end;

double cpu\_time\_used=0;

int rev=0;

int X[9]={18, 523, 301, 1234, 2, 14, 108, 150, 1928};

printf("Input Array: 18\t523\t301\t1234\t2\t14\t108\t150\t1928\nOutpt

Array: ");

start = clock();

Sleep(10);

#pragma omp for

for (int j=0;j<9;j++){

for (int k=X[j];k>0;k=k/10){

rev=(rev\*10)+(k%10);}

printf("%d\t",rev);

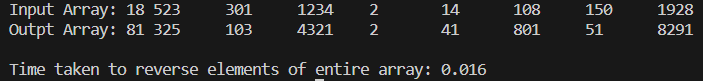
rev=0;}

end = clock();

cpu\_time\_used=cpu\_time\_used +((double) (end - start)) / CLOCKS\_PER\_SEC;

printf("\n\nTime taken to reverse elements of entire array: %0.3f\n",cpu\_time\_used);

return 0;}



2) Write a program in C to simulate all the operations of a calculator. Given inputs A and B, find the output for A+B, A-B, A\*B and A/B.

#include <stdio.h>

#include <omp.h>

#include <time.h>

#include<windows.h>

int main(){

clock\_t start, end;

double cpu\_time\_used=0;

int A,B;

printf("Enter A: \n");

scanf("%d",&A);

printf("Enter B: \n");

scanf("%d",&B);

start = clock();Sleep(10);

#pragma omp parallel num\_threads(4)

{

printf("A+B: %d\n",A+B);

printf("A-B: %d\n",A-B);

printf("A\*B: %d\n",A\*B);

printf("A/B: %d\n",A/B);

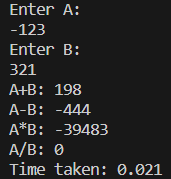
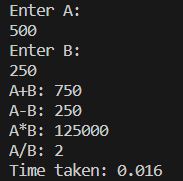
}

end = clock();

printf("\nTotal time taken: %0.3f\n",cpu\_time\_used);

return 0;

}



3) Write a program in C to toggle the character of a given string. Example: suppose the string is “HeLLo”, then the output should be “hEllO”.

#include <stdio.h>

#include <string>

#include <omp.h>

#include <time.h>

#include<windows.h>

int main(){

char str[50],newstr[50];

clock\_t start, end;

double cpu\_time\_used=0;

printf("Enter your text: \n");

scanf("%s", str);

start = clock();Sleep(10);

for (int i=0;i<strlen(str);i++){

#pragma omp parallel num\_threads(3)

{

if (islower(str[i])){

newstr[i]=toupper(str[i]);

}

if (isupper(str[i])){

newstr[i]=tolower(str[i]);}}}

newstr[strlen(str)] = '\0';

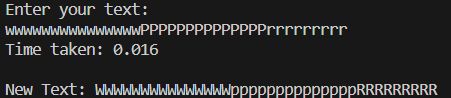
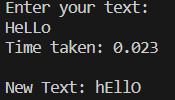
end = clock();

cpu\_time\_used=cpu\_time\_used +((double) (end - start)) / CLOCKS\_PER\_SEC;

printf("Time taken: %0.3f\n",((double) (end - start)) / CLOCKS\_PER\_SEC);

printf("\nNew Text: %s\n",newstr);

return 0;}



4) Write a C program to read a word of length N and produce the pattern as shown in the example. Example: Input: PCBD Output: PCCBBBDDDD

#include <stdio.h>

#include <string.h>

#include <time.h>

#include<omp.h>

int main() {

clock\_t start, end;

double cpu\_time\_used = 0;

int N;

printf("Enter N: \n");

scanf("%d", &N);

char str[N], newstr[N \* (N + 1) / 2 + 1];

printf("Enter your text: ");

scanf("%s", str);

start = clock();

int k = 0;

#pragma omp parallel

for (int i = 0; i < strlen(str); i++) {

for (int j = 0; j <= i; j++) {

newstr[k++] = str[i];}}

newstr[k] = '\0';

end = clock();

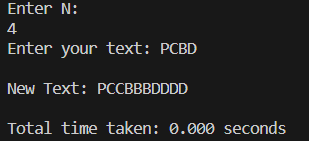
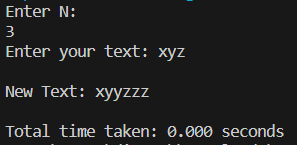
cpu\_time\_used += ((double)(end - start)) / CLOCKS\_PER\_SEC;

printf("\nNew Text: %s\n", newstr);

printf("\nTotal time taken: %0.3f seconds\n", cpu\_time\_used);

return 0;

}



5) Write a C program to read two strings S1 and S2 of same length and produce the resultant string as shown below. S1: string S2: length Resultant String: slternigntgh

#include <stdio.h>

#include <string.h>

#include <time.h>

#include <windows.h>

int main(){

int N;clock\_t start, end;

double cpu\_time\_used=0;

printf("Enter N: \n");

scanf("%d", &N);

char str1[N],str2[N];

printf("Enter S1: \n");

scanf("%s", str1);

printf("Enter S2: \n");

scanf("%s", str2);

printf("Result string: \n");

start = clock();Sleep(10);

#pragma omp for

for (int i=0;i<N;i++){

printf("%c%c",str1[i],str2[i]);

}

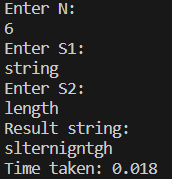
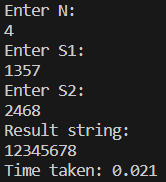
end = clock();

cpu\_time\_used=cpu\_time\_used +((double) (end - start)) / CLOCKS\_PER\_SEC;

printf("\nTime taken: %0.3f\n",cpu\_time\_used);

return 0;

}



6) Write a C program to perform Matrix times vector product operation.

#include <stdio.h>

#include <time.h>

#include <windows.h>

#include <omp.h>

#define MAX\_VALUE 100

void generate\_matrix(int\*\* matrix, int rows, int cols) {

int i, j;

srand(time(NULL));

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

matrix[i][j] = rand() % MAX\_VALUE;

}

}

}

void print\_matrix(int\*\* matrix, int rows, int cols) {

printf("\nMatrix:\n");

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

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

}printf("\n");}}

void generate\_array(int\* a, int size)

{

int i = 0;

srand(time(NULL));

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

{a[i] = rand() % MAX\_VALUE;}}

void print\_array(int\* a, int size) {

printf("\nVector:\n");

for (int i = 0; i < size; i++) {

printf("%d ", a[i]);}printf("\n");}

int main(){

int m,n;clock\_t start, end;

double cpu\_time\_used=0;

printf("Enter m: \n");

scanf("%d", &m);

printf("Enter n: \n");

scanf("%d", &n);

int \*\*mat = (int \*\*)malloc(m \* sizeof(int \*));

for (int i = 0; i < m; i++) {

mat[i] = (int \*)malloc(n \* sizeof(int));}

generate\_matrix(mat, m, n);

print\_matrix(mat, m, n);

int \*vector = (int\*)calloc(n, sizeof(int));

generate\_array(vector,n);

print\_array(vector,n);

printf("\nResult matrix: \n");

start = clock();Sleep(10);

int sum=0;

#pragma omp parallel

for (int i=0;i<m;i++){

for (int j=0;j<n;j++){

sum=sum+(mat[i][j]\*vector[j]);

}

printf("%d\n",sum);

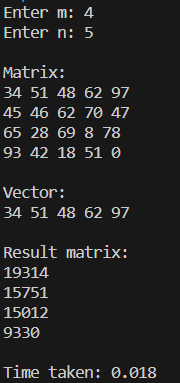
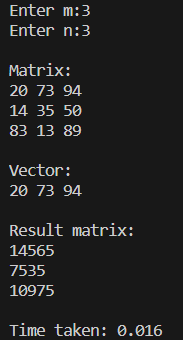
sum=0;

}

end = clock();cpu\_time\_used=cpu\_time\_used +((double) (end - start)) / CLOCKS\_PER\_SEC;

printf("\nTime taken: %0.3f\n",cpu\_time\_used);

return 0;}



7) Write a C program to read a matrix A of size 5x5. It produces a resultant matrix B of size 5x5. It sets all the principal diagonal elements of B matrix with 0. It replaces each row elements in the B matrix in the following manner. If the element is below the principal diagonal it replaces it with the maximum value of the row in the A matrix having the same row number of B. If the element is above the principal diagonal it replaces it with the minimum value of the row in the A matrix having the same row number of B.

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <omp.h>

#define MAX\_VALUE 100

void generate\_matrix(int\*\* matrix, int rows, int cols) {

srand(time(NULL));

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

matrix[i][j] = rand() % MAX\_VALUE;}}}

void print\_matrix(int\*\* matrix, int rows, int cols) {

printf("\nMatrix:\n");

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

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

void processMatrix(int\*\* A, int\*\* B, int rows, int cols) {

#pragma parallel for collapsed(2)

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

if (j == i) {

B[i][j] = 0;

} else if (j > i) {

int maxVal = A[i][0];

for (int k = 1; k < cols; k++) {

if (A[i][k] > maxVal) {

maxVal = A[i][k];}}

B[i][j] = maxVal;

} else {

int minVal = A[i][0];

for (int k = 1; k < cols; k++) {

if (A[i][k] < minVal) {

minVal = A[i][k];}}

B[i][j] = minVal;}}}}

int main() {

int rows, cols;clock\_t start, end;

double cpu\_time\_used=0;

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

scanf("%d", &rows);

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

scanf("%d", &cols);

start = clock();

int \*\*A = (int \*\*)malloc(rows \* sizeof(int \*));

int \*\*B = (int \*\*)malloc(rows \* sizeof(int \*));

for (int i = 0; i < rows; i++) {

A[i] = (int \*)malloc(cols \* sizeof(int));

B[i] = (int \*)malloc(cols \* sizeof(int));}

generate\_matrix(A, rows, cols);

printf("\nMatrix A:\n");

print\_matrix(A, rows, cols);

processMatrix(A, B, rows, cols);

printf("\nMatrix B:\n");

print\_matrix(B, rows, cols);

for (int i = 0; i < rows; i++) {

free(A[i]);free(B[i]);

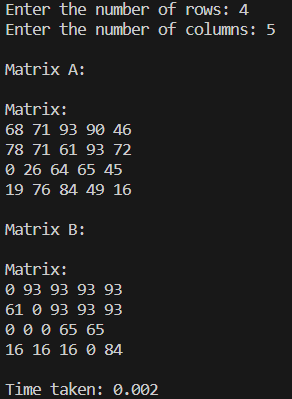
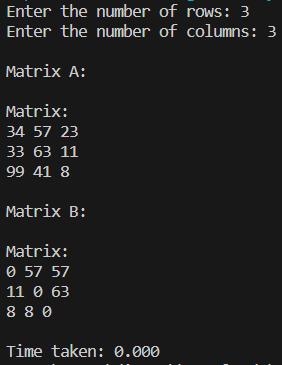
}free(A);free(B);

end = clock();

cpu\_time\_used=cpu\_time\_used +((double) (end - start)) / CLOCKS\_PER\_SEC;

printf("\nTime taken: %0.3f\n",cpu\_time\_used);

return 0;}



8) Write a C program that reads a matrix of size MxN and produce an output matrix B of same size such that it replaces all the non-border elements of A with its equivalent 1’s complement and remaining elements same as matrix A. Also produce a matrix D as shown below.

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <omp.h>

#define MAX\_VALUE 100

void decToBinary(int n) {

for (int i = 31; i >= 0; i--) {

int k = n >> i;

if (k & 1)

printf("1");

else

printf("0");

}

}

int onesComplement(int num) {

return ~num;

}

void processMatrix(int \*\*A, int \*\*B, int \*\*D, int ROWS, int COLS) {

#pragma omp for collapsed(2)

for (int i = 0; i < ROWS; i++) {

for (int j = 0; j < COLS; j++) {

if (i != 0 && j != 0 && i != ROWS - 1 && j != COLS - 1) {

B[i][j] = onesComplement(A[i][j]);

decToBinary(B[i][j]);

} else {

B[i][j] = A[i][j];

D[i][j] = A[i][j];

}

}

}

}

void generate\_matrix(int \*\*matrix, int rows, int cols) {

srand(time(NULL));

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

matrix[i][j] = rand() % MAX\_VALUE;

}

}

}

void print\_matrix(int \*\*matrix, int rows, int cols) {

printf("\nMatrix:\n");

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

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

}

printf("\n");

}

}

int main() {

clock\_t start, end;

double cpu\_time\_used = 0;

int ROWS, COLS;

printf("Enter number of rows: ");

scanf("%d", &ROWS);

printf("Enter number of columns: ");

scanf("%d", &COLS);

int \*\*A = (int \*\*)malloc(ROWS \* sizeof(int \*));

int \*\*B = (int \*\*)malloc(ROWS \* sizeof(int \*));

int \*\*D = (int \*\*)malloc(ROWS \* sizeof(int \*));

for (int i = 0; i < ROWS; i++) {

A[i] = (int \*)malloc(COLS \* sizeof(int));

B[i] = (int \*)malloc(COLS \* sizeof(int));

D[i] = (int \*)malloc(COLS \* sizeof(int));

}

generate\_matrix(A, ROWS, COLS);

printf("\nMatrix before processing:\n");

print\_matrix(A, ROWS, COLS);

start = clock();

processMatrix(A, B, D, ROWS, COLS);

printf("\nMatrix B after processing:\n");

print\_matrix(B, ROWS, COLS);

printf("\nMatrix D after processing:\n");

print\_matrix(D, ROWS, COLS);

end = clock();

cpu\_time\_used = ((double)(end - start)) / CLOCKS\_PER\_SEC;

printf("Time taken: %0.3f\n", cpu\_time\_used);

for (int i = 0; i < ROWS; i++) {

free(A[i]);

free(B[i]);

free(D[i]);

}

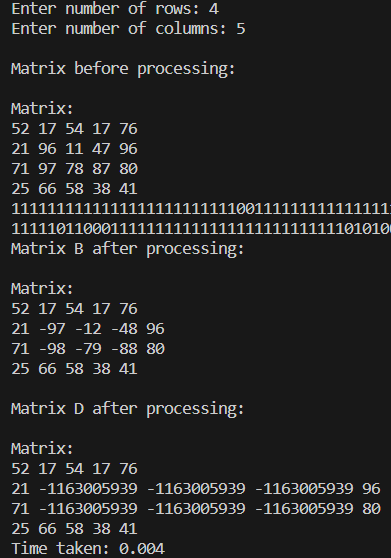
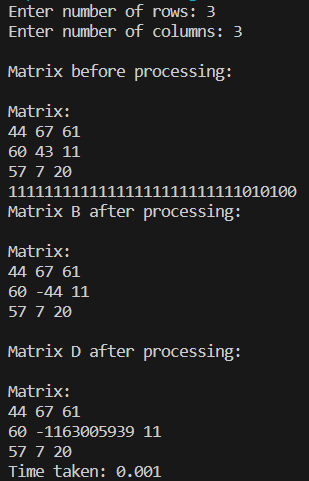
free(A);

free(B);

free(D);

return 0;

}



9) Write a C program that reads a character type matrix and integer type matrix B of size MxN. It produces and output string STR such that, every character of A is repeated r times (where r is the integer value in matrix B which is having the same index as that of the character taken in A).

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <omp.h>

int main() {

clock\_t start, end;

double cpu\_time\_used = 0;

int m, n;

printf("Enter m: ");

scanf("%d", &m);

printf("Enter n: ");

scanf("%d", &n);

char A[m][n];

int B[m][n];

start = clock();

srand(time(NULL));

printf("\nMatrix A:\n");

for (int i = 0; i < m; i++) {

for (int j = 0; j < n; j++) {

A[i][j] = 'A' + rand() % 26;

printf("%c ", A[i][j]);

}

printf("\n");

}

printf("\nMatrix B:\n");

for (int i = 0; i < m; i++) {

for (int j = 0; j < n; j++) {

B[i][j] = rand() % 5;

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

}

printf("\n");

}

printf("\nOutput string STR: ");

#pragma omp parallel for collapsed(3)

for (int i = 0; i < m; i++) {

for (int j = 0; j < n; j++) {

for (int k = 0; k < B[i][j]; k++) {

printf("%c", A[i][j]);

}

}

}

printf("\n");

end = clock();

cpu\_time\_used = ((double)(end - start)) / CLOCKS\_PER\_SEC;

printf("Time taken: %0.3f\n", cpu\_time\_used);

return 0;

}

