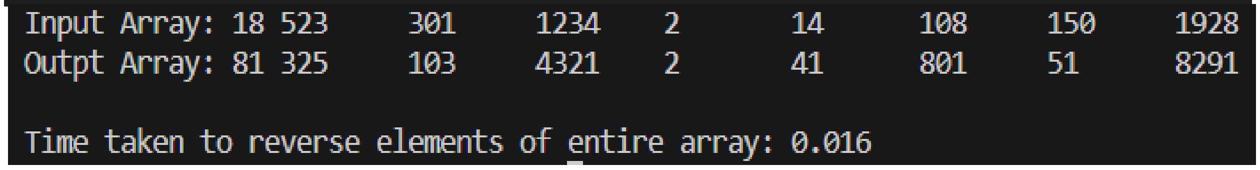
PP LAB WEEK-2

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

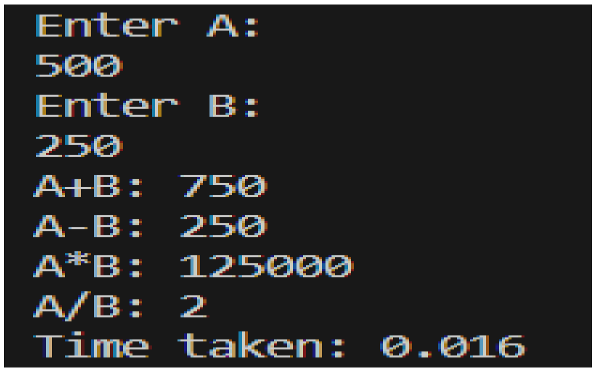
```c
#include <stdio.h>
#include <omp.h>
#include <time.h>
#include <windows.h>
int main(){
    clock_t start, end;
    double cpu_time_used=0;
    int rev=0;
    int X[9]={18, 523, 301, 1234, 2, 14, 108, 150, 1928};
printf("Input Array: 18\t523\t301\t1234\t2\t14\t108\t150\t1928\nOutpt
Array: ");
    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;}
```



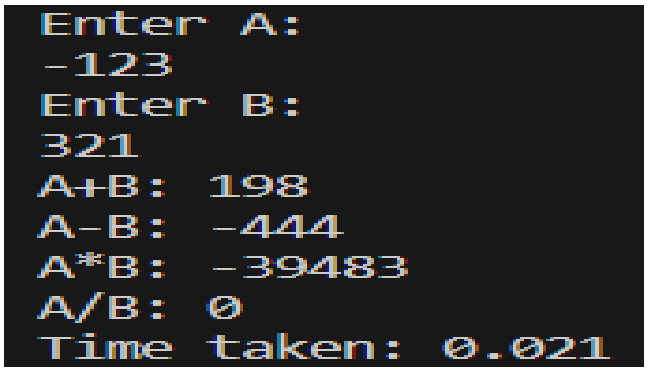
```
Input Array: 18 523     301     1234    2       14      108     150     1928
Outpt Array: 81 325     103     4321    2       41      801     51      8291

Time taken to reverse elements of entire array: 0.016
```

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.

```c
#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;
}
```



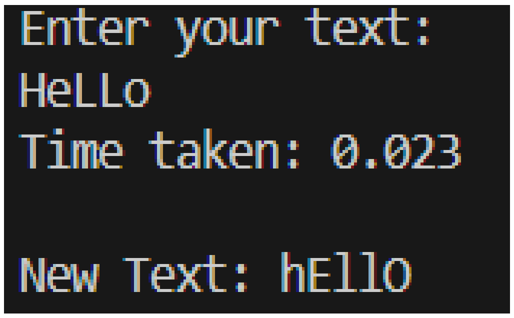
```
Enter A:
500
Enter B:
250
A+B: 750
A-B: 250
A*B: 125000
A/B: 2
Time taken: 0.016
```



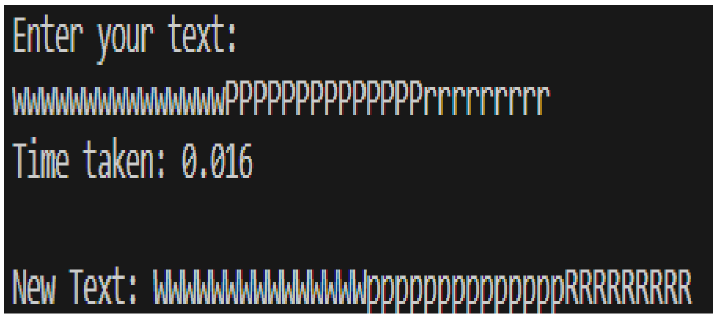
```
Enter A:
-123
Enter B:
321
A+B: 198
A-B: -444
A*B: -39483
A/B: 0
Time taken: 0.021
```

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”.

```c
#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;}
```



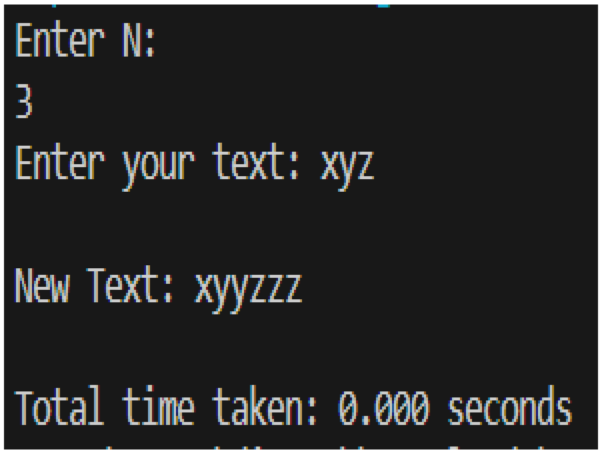
```
Enter your text:
HeLLo
Time taken: 0.023

New Text: hEllo
```



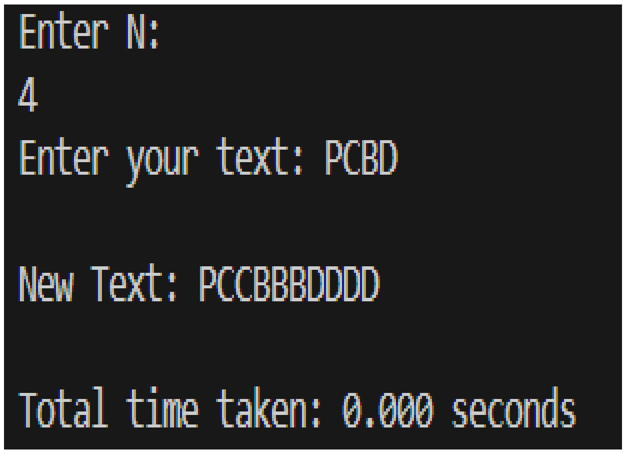
```
Enter your text:
wwwwwwwwwwwwwwwPPPPPPPPPPPPPPrrrrrrrrr
Time taken: 0.016


New Text: WWWWWWWWWWWWWWWppppppppppppppRRRRRRRRR
```

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

```c
#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;
}
```



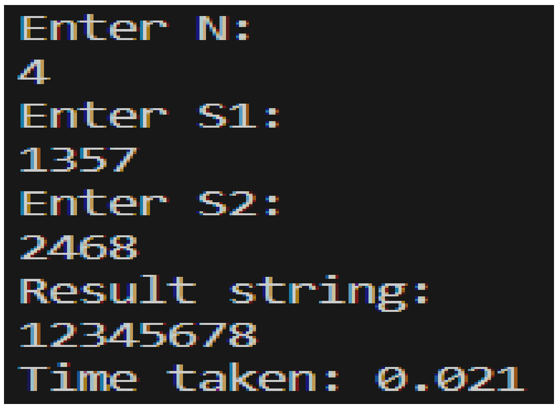
```
Enter N:
3
Enter your text: xyz

New Text: xyyzzz

Total time taken: 0.000 seconds
```



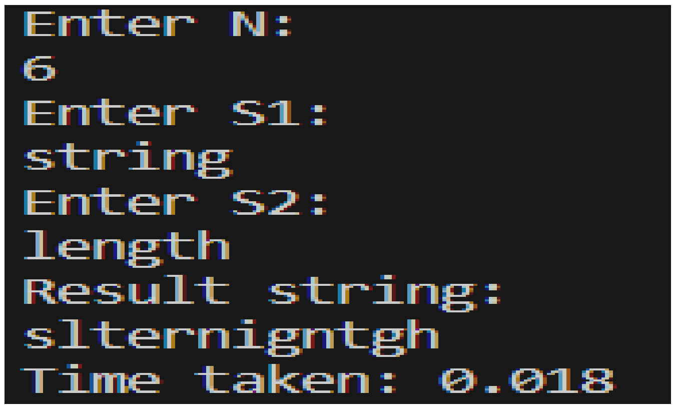
```
Enter N:
4
Enter your text: PCBD

New Text: PCCBBBDDDD

Total time taken: 0.000 seconds
```

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

```c
#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;
}
```



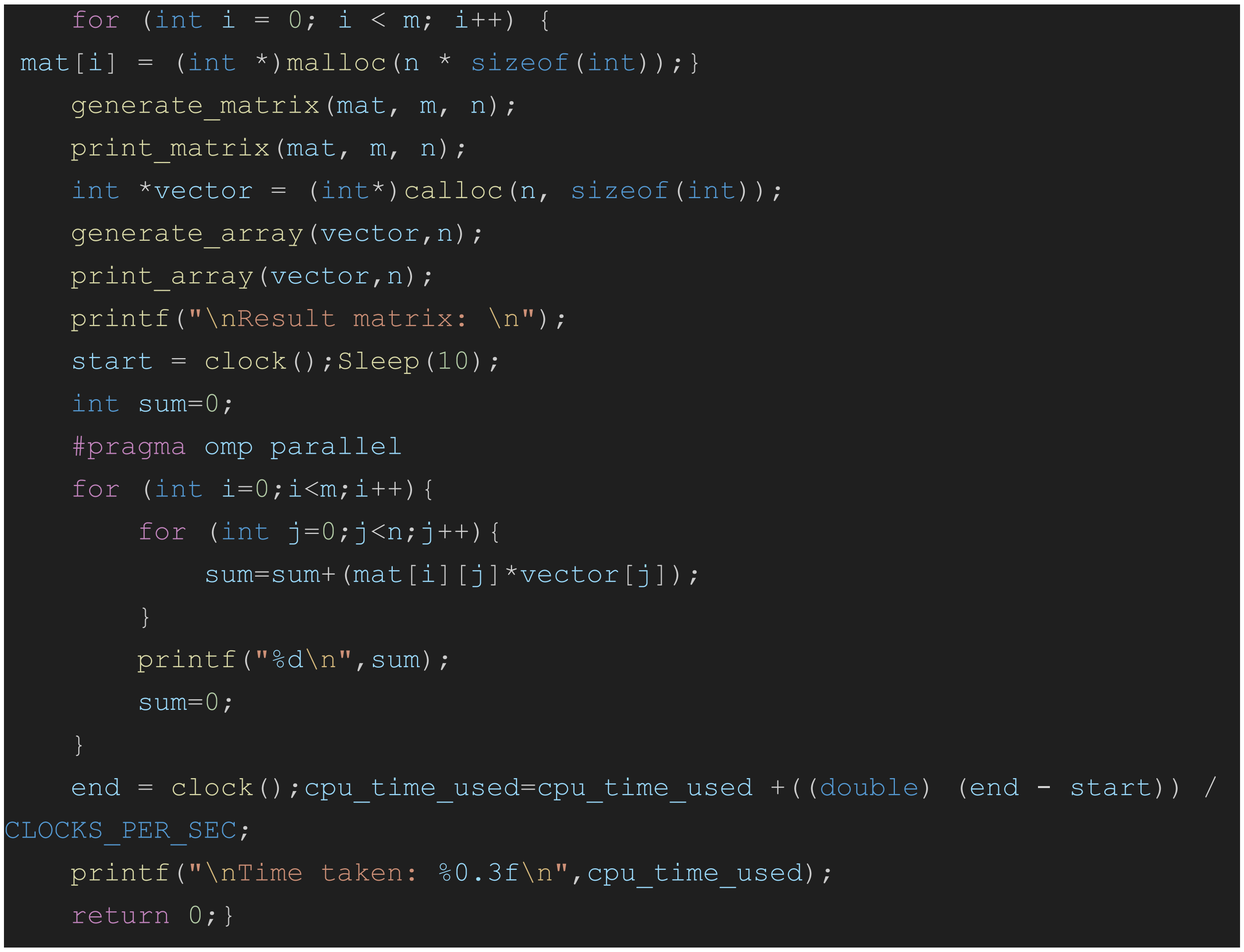
```
Enter N:
4
Enter S1:
1357
Enter S2:
2468
Result string:
12345678
Time taken: 0.021
```



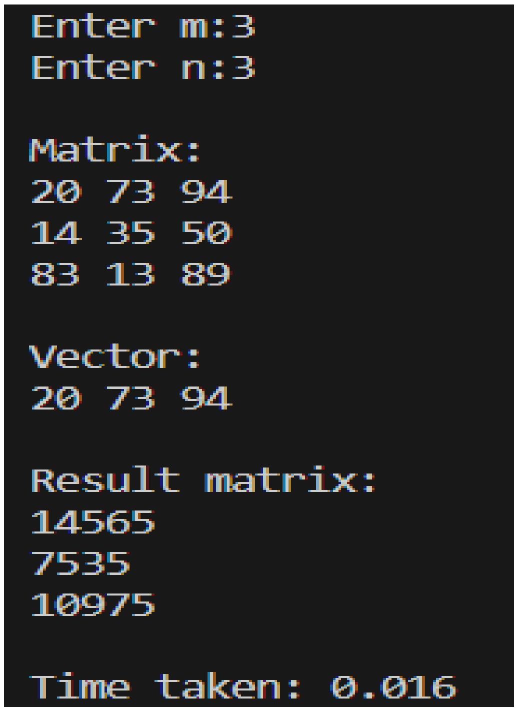
```
Enter N:
6
Enter S1:
string
Enter S2:
length
Result string:
slternigntgh
Time taken: 0.018
```

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

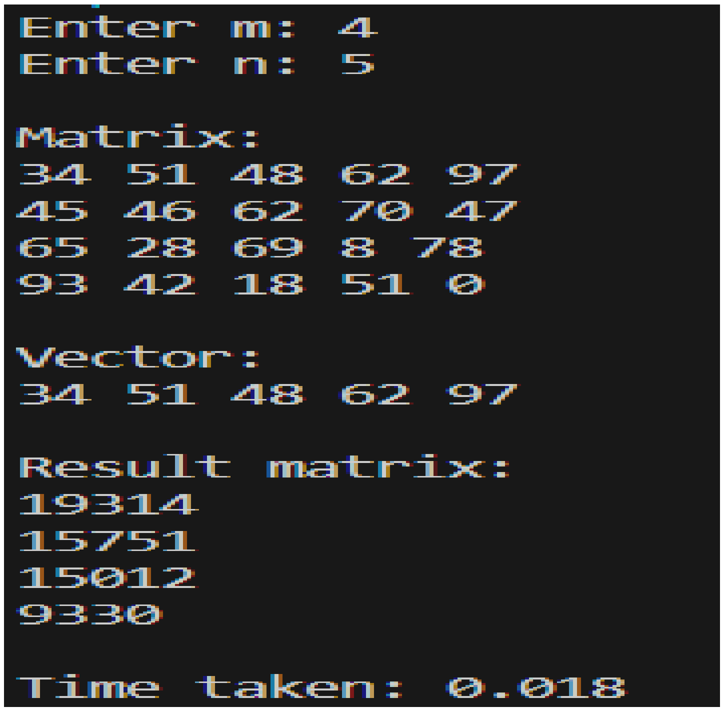
```
#include <stdio.h>
#include <time.h>
#include <windows.h>
#include <omp.h>
#define MAX_VALUE 100
void generate_matrix(int** matrix, int rows, int cols) {
    int i, j;
    srand(time(NULL));
    for (i = 0; i < rows; i++) {
        for (j = 0; j < cols; j++) {
            matrix[i][j] = rand() % 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;}
```



```
Enter m:3
Enter n:3

Matrix:
20 73 94
14 35 50
83 13 89

Vector:
20 73 94

Result matrix:
14565
7535
10975

Time taken: 0.016
```

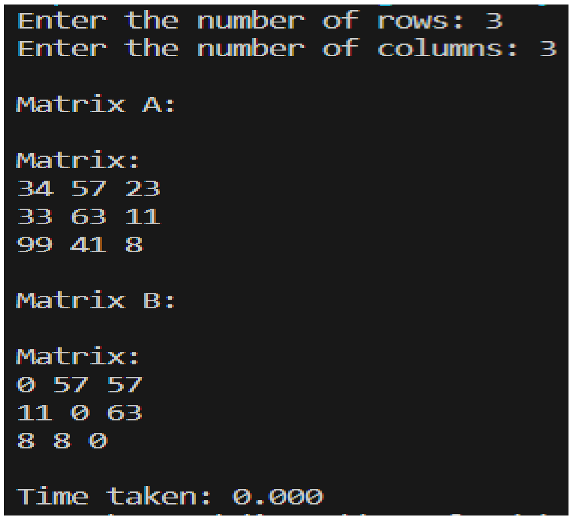


```
Enter m: 4
Enter n: 5

Matrix:
34 51 48 62 97
45 46 62 70 47
65 28 69 8 78
93 42 18 51 0

Vector:
34 51 48 62 97

Result matrix:
19314
15751
15012
9330

Time taken: 0.018
```

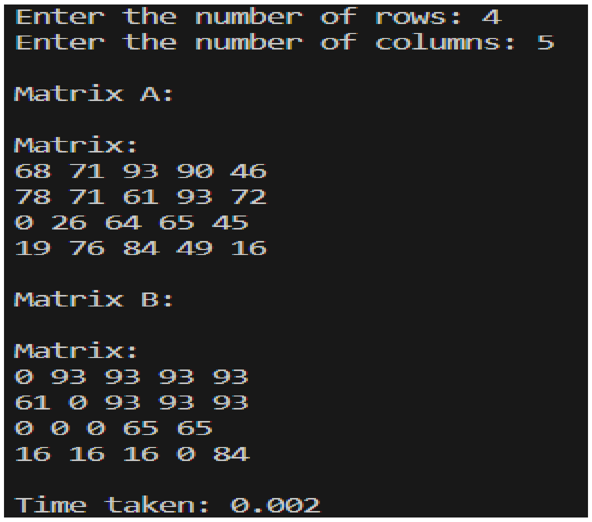
7) Write a C program to read a matrix A of size 5x5. It produces a resultant matrix B of size 5x5. It sets all the principal diagonal elements of B matrix with 0. It replaces each row elements in the B matrix in the following manner. If the element is below the principal diagonal it replaces it with 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() {
```

```c
    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;}
```



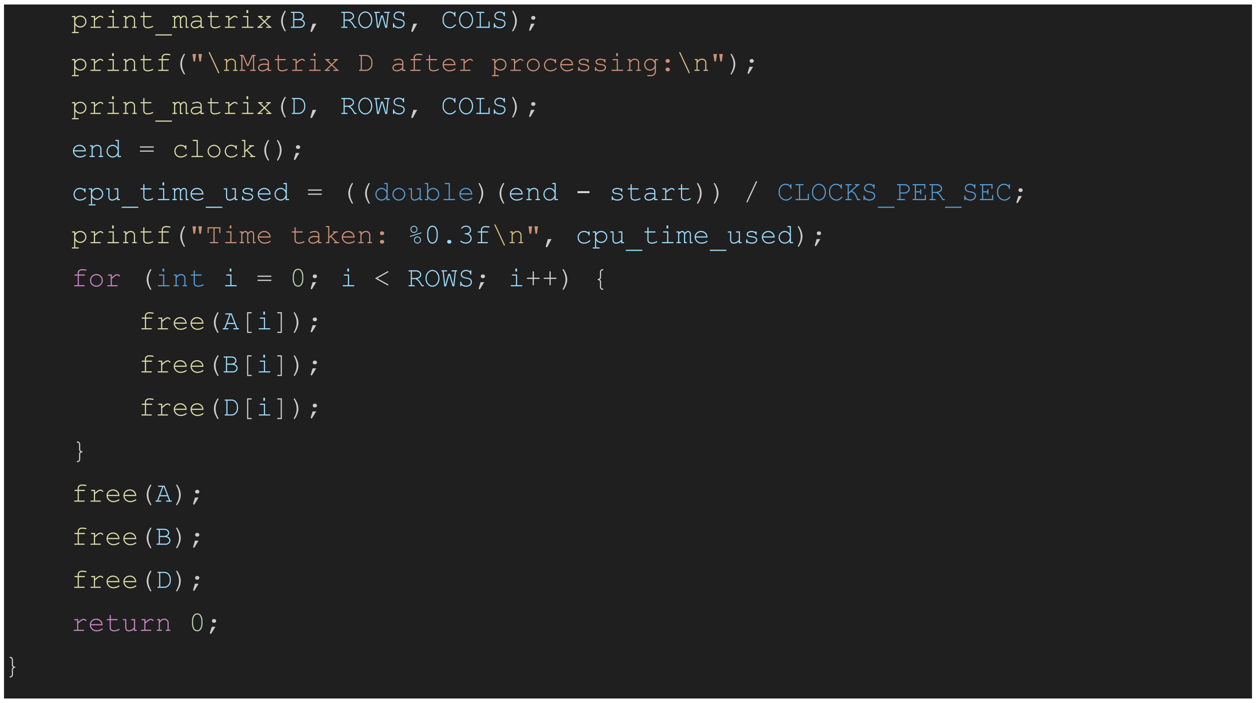
```
Enter the number of rows: 3
Enter the number of columns: 3

Matrix A:

Matrix:
34 57 23
33 63 11
99 41 8

Matrix B:

Matrix:
0 57 57
11 0 63
8 8 0

Time taken: 0.000
```



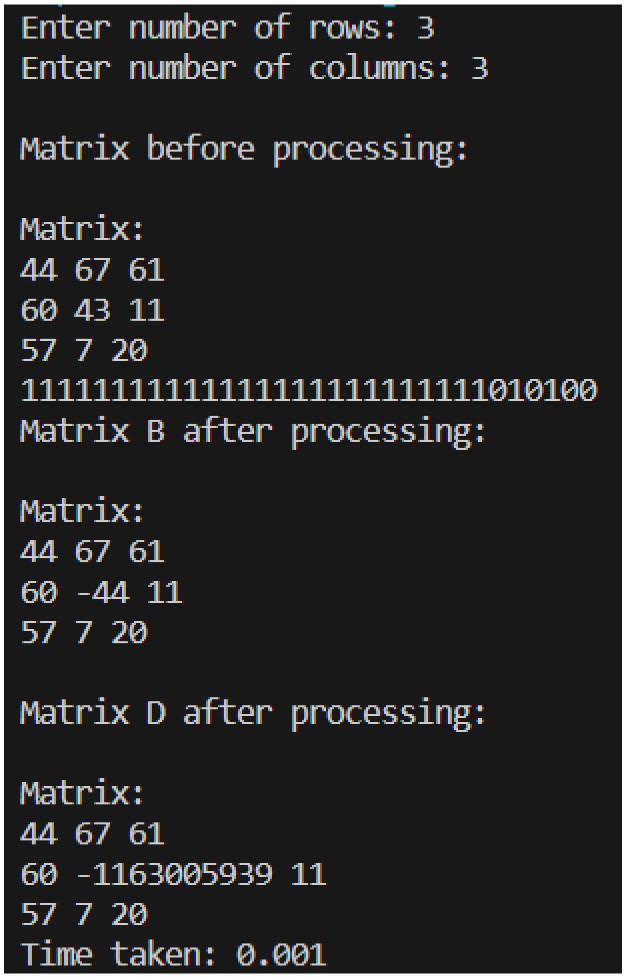
```
Enter the number of rows: 4
Enter the number of columns: 5

Matrix A:

Matrix:
68 71 93 90 46
78 71 61 93 72
0 26 64 65 45
19 76 84 49 16

Matrix B:

Matrix:
0 93 93 93 93
61 0 93 93 93
0 0 0 65 65
16 16 16 0 84

Time taken: 0.002
```

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.

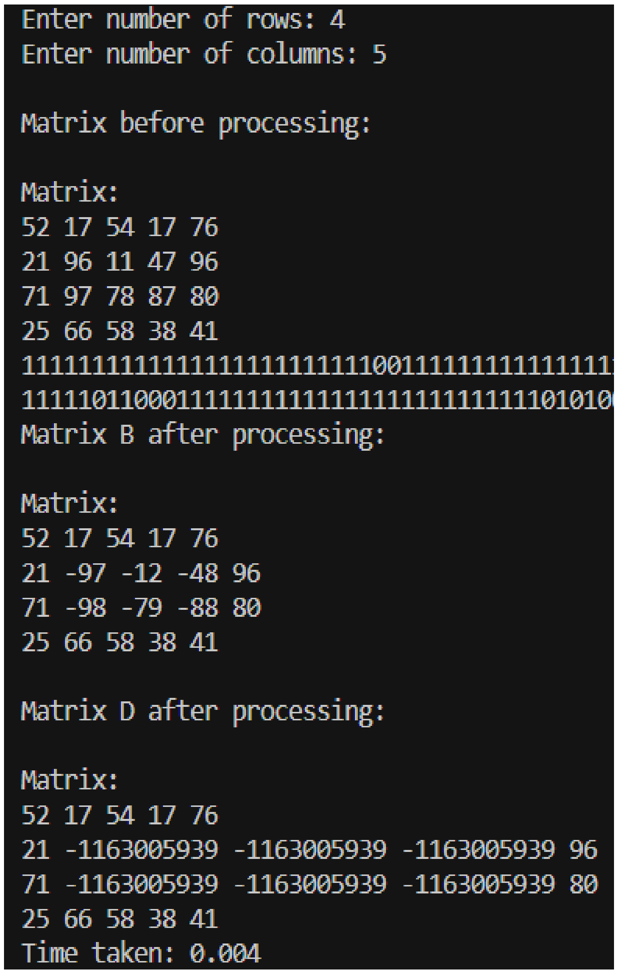
```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <omp.h>

#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");
```



```c
    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;
}
```



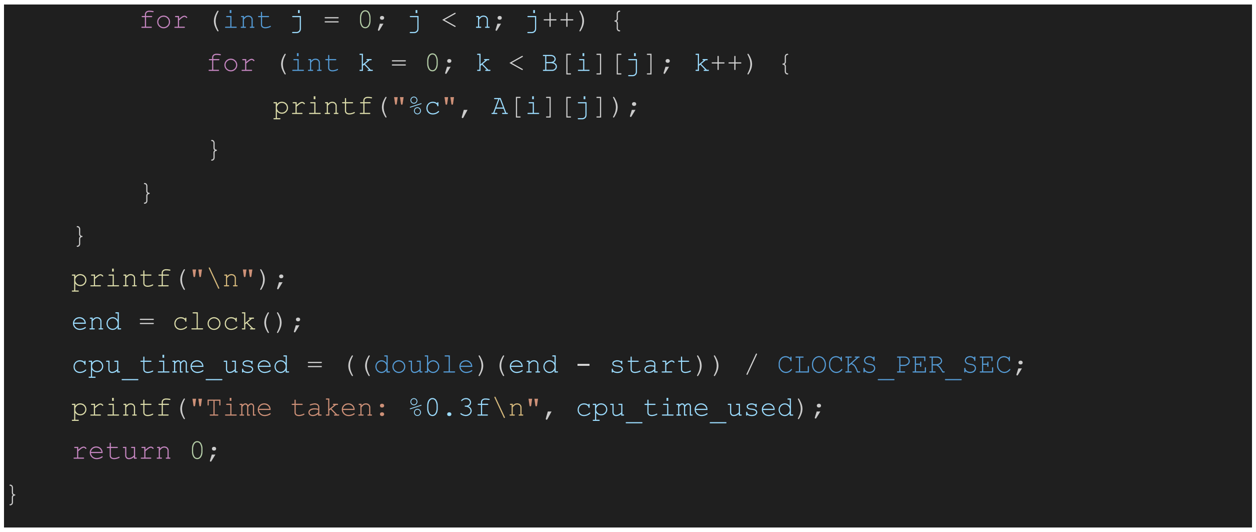
```
Enter number of rows: 3
Enter number of columns: 3

Matrix before processing:

Matrix:
44 67 61
60 43 11
57 7 20
11111111111111111111111111010100
Matrix B after processing:

Matrix:
44 67 61
60 -44 11
57 7 20

Matrix D after processing:

Matrix:
44 67 61
60 -1163005939 11
57 7 20
Time taken: 0.001
```



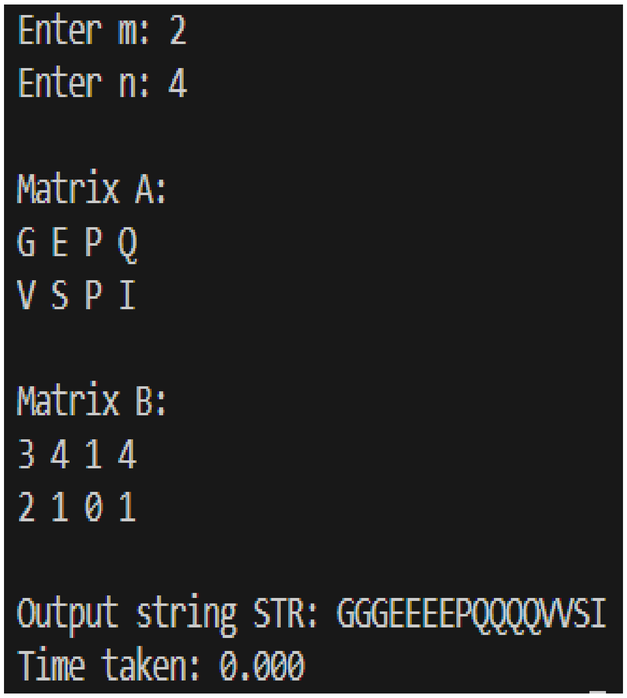
```
Enter number of rows: 4
Enter number of columns: 5

Matrix before processing:

Matrix:
52 17 54 17 76
21 96 11 47 96
71 97 78 87 80
25 66 58 38 41
111111111111111111111111100111111111111111
1111101100011111111111111111111111111010100
Matrix B after processing:

Matrix:
52 17 54 17 76
21 -97 -12 -48 96
71 -98 -79 -88 80
25 66 58 38 41

Matrix D after processing:

Matrix:
52 17 54 17 76
21 -1163005939 -1163005939 -1163005939 96
71 -1163005939 -1163005939 -1163005939 80
25 66 58 38 41
Time taken: 0.004
```

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).

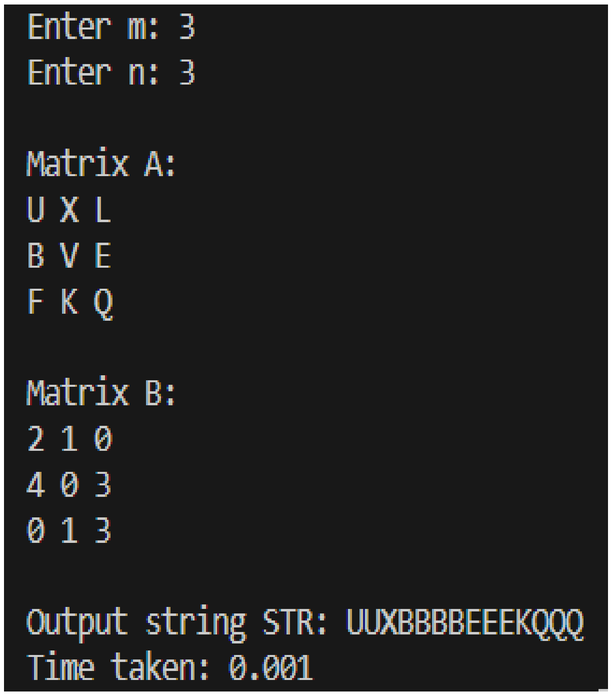
```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <omp.h>
int main() {
    clock_t start, end;
    double cpu_time_used = 0;
    int m, n;
    printf("Enter m: ");
    scanf("%d", &m);
    printf("Enter n: ");
    scanf("%d", &n);
    char A[m][n];
    int B[m][n];
    start = clock();
    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++) {
```



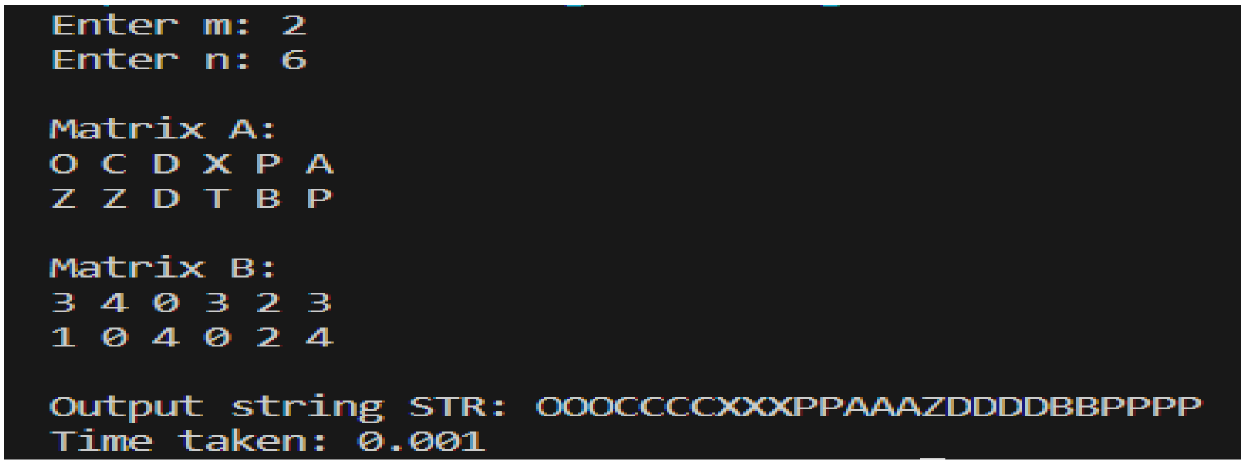
```
        for (int j = 0; j < n; j++) {
            for (int k = 0; k < B[i][j]; k++) {
                printf("%c", A[i][j]);
            }
        }
    }
    printf("\n");
    end = clock();
    cpu_time_used = ((double)(end - start)) / CLOCKS_PER_SEC;
    printf("Time taken: %0.3f\n", cpu_time_used);
    return 0;
}
```



```
Enter m: 2
Enter n: 4

Matrix A:
G E P Q
V S P I

Matrix B:
3 4 1 4
2 1 0 1

Output string STR: GGGEEEEPQQQQVVSI
Time taken: 0.000
```



```
Enter m: 3
Enter n: 3

Matrix A:
U X L
B V E
F K Q

Matrix B:
2 1 0
4 0 3
0 1 3

Output string STR: UUXBBBBEEEKQQQ
Time taken: 0.001
```



```
Enter m: 2
Enter n: 6

Matrix A:
O C D X P A
Z Z D T B P

Matrix B:
3 4 0 3 2 3
1 0 4 0 2 4

Output string STR: OOOCCCCXXXPPAAAZDDDDBBPPPP
Time taken: 0.001
```