PROGRAMMING LAB ASSIGNMENTS (C, C++)

MCA, SEMESTER-1

ARKA DAS, 002210503046
NAIMUR RAHAMAN, 002210503040
PURNENDU MANNA, 002210503043

SET-1

Question 13:

Problem Statement:

Consider that M is an n x n matrix whose each row contains real numbers or 0 such that the sum of each row is 1. If R is an n-dimensional column vector whose each component is 1/n. Then use random number generator to create the matrix M. Write a program to compute: $R = (M^p)R$, where p should be takes an input.

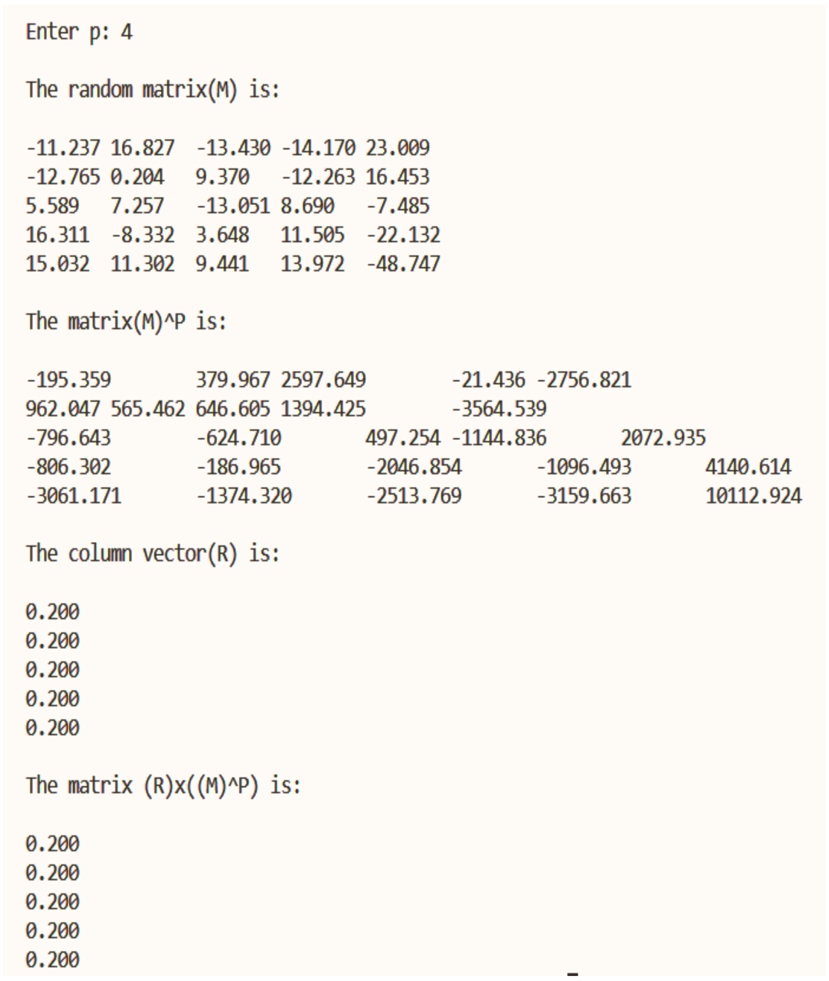
Show that for any positive integer P the relation $R = (M^p)R$ holds.

Source Code:

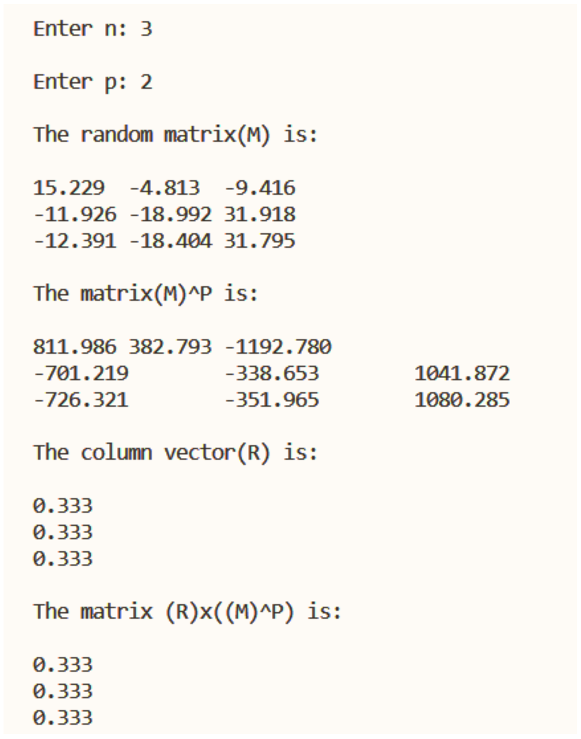
```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void createMatrix(float **, int );
void multiplyMatrix(float **, int , int , float **, int , int , float **);
void display(float **, int , int );

int main()
{
    int n, p, i, j;
    float **matrix, **r, **mPowerP, **final;
    srand(time(0));

    printf("\nEnter n: ");
    scanf("%d", &n);
    printf("\nEnter p: ");
    scanf("%d", &p);

    //this is the matrix which will be randomly filled with data
    matrix = (float** )malloc(sizeof(float* ) * n);
    for(i=0;i<n;i++)
        matrix[i] = (float* )malloc(sizeof(float ) * n);

    //this mPowerP holds the result of m^p operation
    mPowerP = (float** )malloc(sizeof(float* ) * n);
    for(i=0;i<n;i++)
        mPowerP[i] = (float* )malloc(sizeof(float ) * n);
    for(i=0;i<n;i++)     //initializing it to 0
        for(j=0;j<n;j++)
            *(*(mPowerP + i) + j) = 0.f;

    //r is the n-dimensional column vector. r is (1 X n)
    r = (float** )malloc(sizeof(float* ) * n);
    for(i=0;i<n;i++)
        r[i] = (float* )malloc(sizeof(float ) * 1);
    //final is the matrix which stores the ultimate result
    //which is - r * (m^p)
    final = (float** )malloc(sizeof(float* ) * n);
    for(i=0;i<n;i++)
        final[i] = (float* )malloc(sizeof(float ) * 1);

    //initializing r and final. Each element of r is 1/n
    for(i=0;i<n;i++) {
        for(j=0;j<1;j++){
            *(*(r + i) + j) = (float)1/n;
            *(*(final + i) + j) = 0;
        }
    }

    createMatrix(matrix, n);
    printf("\nThe random matrix(M) is:\n");
    display(matrix, n, n);

    for(i=1;i<=p;i++)
        multiplyMatrix(matrix, n, n, matrix, n, n, mPowerP);
    printf("\n\nThe matrix(M)^P is:\n");
    display(mPowerP, n, n);

    printf("\n\nThe column vector(R) is:\n");
    display(r, n, 1);

    printf("\n\nThe matrix (R)x((M)^P) is:\n");
    multiplyMatrix(matrix, n, n, r, n, 1, final);
    display(final, n, 1);

}

void multiplyMatrix(float **matA, int r1, int c1, float **matB, int r2,
int c2, float **res) {
    int i, j, k;
    for (i = 0; i < r1; ++i) {
        for (j = 0; j < c2; ++j) {
            for (k = 0; k < c1; ++k) {
                *(*(res + i) + j) += *(*(matA + i) + k) * *(*(matB + k) +
j);
            }
        }
    }
}

void createMatrix(float **matrix, int n) {
    int i, j;
    for(i=0;i<n;i++) {
        float sum = 0;
        for(j=0;j<n-1;j++) {
            float val = ((float) rand() / (float)(RAND_MAX));
            int nVal = rand()%20;
            int sign = (rand()%2 == 0) ? -1 : 1;
```

```
                val = sign*(val + nVal);
                sum += val;
                *(*(matrix + i) + j) = val;
            }
            *(*(matrix + i) + (n-1)) = 1 - sum;
        }

}

void display(float **matrix, int n, int m) {
    int i, j;
    for(i=0;i<n;i++) {
        printf("\n");
        for(j=0;j<m;j++)
            printf("%.3f\t", *(*(matrix + i) + j));
    }
}
```

Output:



```
Enter p: 4

The random matrix(M) is:

-11.237 16.827  -13.430 -14.170 23.009
-12.765 0.204   9.370   -12.263 16.453
5.589   7.257   -13.051 8.690   -7.485
16.311  -8.332  3.648   11.505  -22.132
15.032  11.302  9.441   13.972  -48.747

The matrix(M)^P is:

-195.359        379.967 2597.649        -21.436 -2756.821
962.047 565.462 646.605 1394.425        -3564.539
-796.643        -624.710        497.254 -1144.836       2072.935
-806.302        -186.965        -2046.854       -1096.493       4140.614
-3061.171       -1374.320       -2513.769       -3159.663       10112.924

The column vector(R) is:

0.200
0.200
0.200
0.200
0.200

The matrix (R)x((M)^P) is:

0.200
0.200
0.200
0.200
0.200
```



```
Enter n: 3

Enter p: 2

The random matrix(M) is:

15.229  -4.813  -9.416
-11.926 -18.992 31.918
-12.391 -18.404 31.795

The matrix(M)^P is:

811.986 382.793 -1192.780
-701.219        -338.653        1041.872
-726.321        -351.965        1080.285

The column vector(R) is:

0.333
0.333
0.333

The matrix (R)x((M)^P) is:

0.333
0.333
0.333
```

Question 12:

Problem Statement:

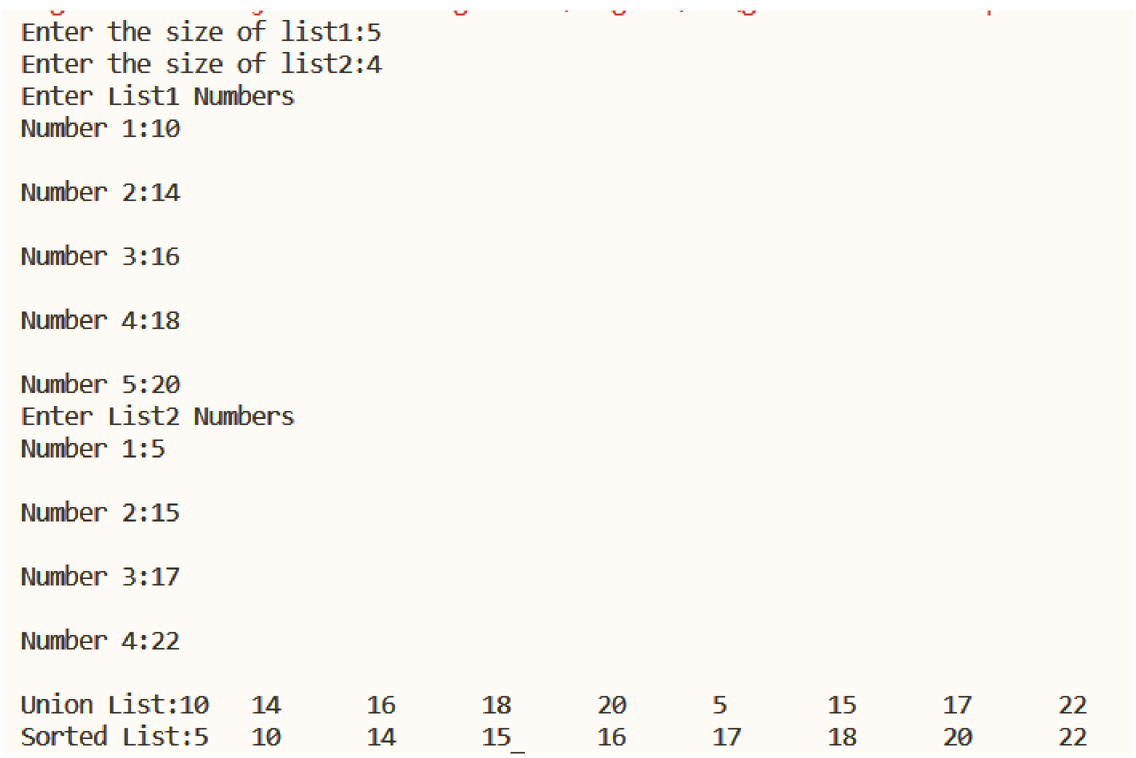
Write a program to compute a union of two sorted lists of integers so that the resultant list remains sorted.

Source Code:

```
#include<stdio.h>

int main()
{
	int list1[10],list2[10],arr[20],i,n1,n2,j=0,temp;

	printf("Enter the size of list1:");
	scanf("%d",&n1);

	printf("Enter the size of list2:");
	scanf("%d",&n2);

	printf("Enter List1 Numbers");
	for(i=0;i<n1;i++)
	{
		printf("\nNumber %d:",i+1);
		scanf("%d",&list1[i]);
		arr[i]=list1[i];
	}

	printf("Enter List2 Numbers");
	for(i=0;i<n2;i++)
	{
		printf("\nNumber %d:",i+1);
		scanf("%d",&list2[i]);
	}

	for(i=n1;i<(n1+n2);i++)
	{
		arr[i]=list2[j];
		j++;
	}

	printf("\nUnion List:");
	for(i=0;i<(n1+n2);i++)
	{
		printf("%d\t",arr[i]);
	}

	for(i=0;i<(n1+n2);i++)
	{
		for(int    j=i+1;j<(n1+n2);j++)
		if(arr[i]>arr[j])
		{
```

```
                    temp=arr[i];
                    arr[i]=arr[j];
                    arr[j]=temp;
                }
            }

        printf("\nSorted List:");
        for(i=0;i<(n1+n2);i++)
        {
                printf("%d\t",arr[i]);
        }

}
```

Output:



```
Enter the size of list1:5
Enter the size of list2:4
Enter List1 Numbers
Number 1:10

Number 2:14

Number 3:16

Number 4:18

Number 5:20
Enter List2 Numbers
Number 1:5

Number 2:15

Number 3:17

Number 4:22

Union List:10   14      16      18      20      5       15      17      22
Sorted List:5   10      14      15_     16      17      18      20      22
```

Question 10:

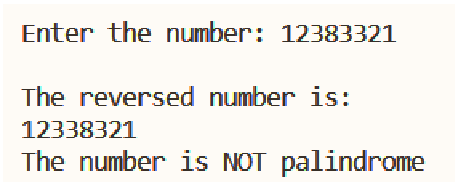
Problem Statement:

Write a program to find the reverse of any number and check whether the number is a palindrome or not.

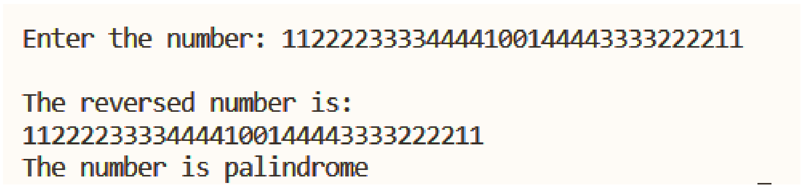
Source Code:

```c
#include <stdio.h>
#include <string.h>

void reverse(char *number, char *reversed, int *size)
{
    int i,j;
    for(i=*size-1,j=0;i>=0;i--,j++)
        *(reversed + j) = *(number + i);
}

void trucn(char *number, int *size)
{
    int i=0,j;
    while(i < *size) {
        if(*(number+i) != '0')
            break;
        i++;
    }

    if(i==0)
        return;

    for(j=i;j<*size;j++)
        *(number+(j-i)) = *(number+j);
    *size = *size - i;
    *(number+(*size)) = '\0';
}

int main()
{
    char number[50] = {' '}, reversed[50] = {' '};
    int i, n = 0;

    printf("\nEnter the number: ");
    scanf("%s", &number);
    n = strlen(number);

    trucn(number, &n);
    reverse(number, reversed, &n);
    trucn(reversed, &n);

    printf("\nThe reversed number is: ");
    printf("\n%s", reversed);
    if(strcmp(number, reversed) == 0)
        printf("\nThe number is palindrome");
    else
        printf("\nThe number is NOT palindrome");

    return 0;
}
```

Output:



```
Enter the number: 12383321

The reversed number is:
12338321
The number is NOT palindrome
```



```
Enter the number: 11222233334444100144443333222211

The reversed number is:
11222233334444100144443333222211
The number is palindrome
```

Question 2:

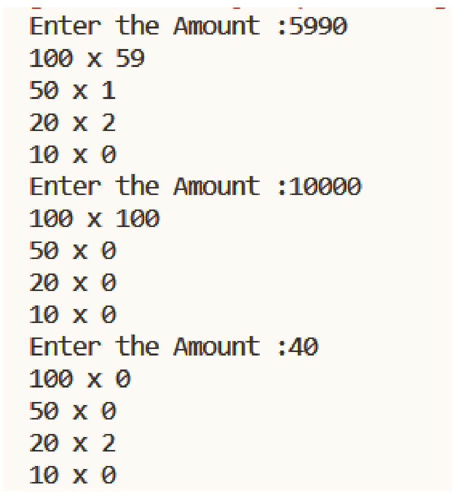
Problem Statement:

In a banking system, there are the following denominations of notes: Rs. 10, Rs. 20. Rs. 50. Rs. 100. Write a program that will accept an amount and find the minimum number of each note required to pay the amount.

Source Code:

```
#include<stdio.h>
int main(){
    int n;

    for(int i = 0 ; i < 6 ; i++){
        printf("Enter the Amount :");
        scanf("%d", &n);
        if(n%10 == 0){
            printf("100 x %d\n", n/100);
            int p= n%100;
            printf("50 x %d\n", p/50);
            p = p % 50;
            printf("20 x %d\n", p/20);
            p = p % 20;
            printf("10 x %d\n", p/10);
        }
        else{
            printf("Not Possible\n");
        }
    }
    return 0;
}
```

Output:



```
Enter the Amount :5990
100 x 59
50 x 1
20 x 2
10 x 0
Enter the Amount :10000
100 x 100
50 x 0
20 x 0
10 x 0
Enter the Amount :40
100 x 0
50 x 0
20 x 2
10 x 0
```

SET-2

Question 5:

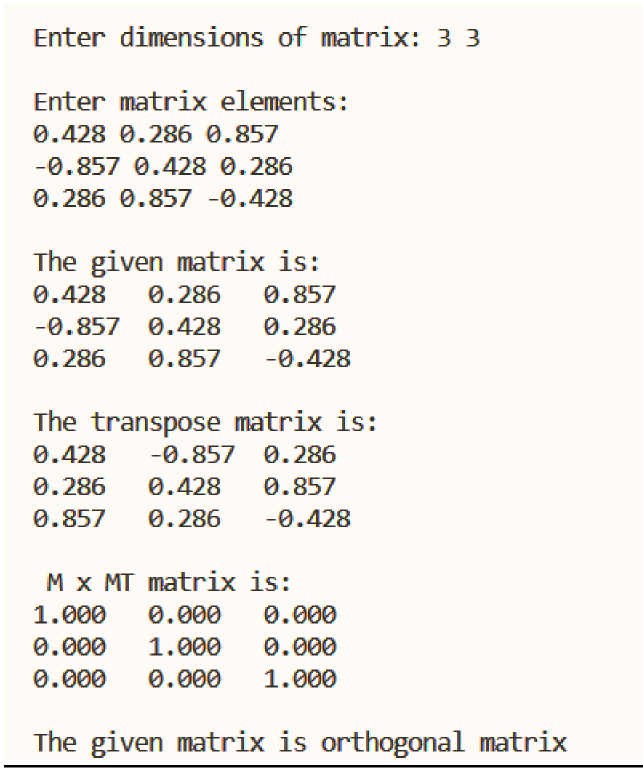
Problem Statement:

Write a function to read a matrix, transpose a matrix, multiply two matrices and use these functions in main() to check whether an input matrix is orthogonal or not.

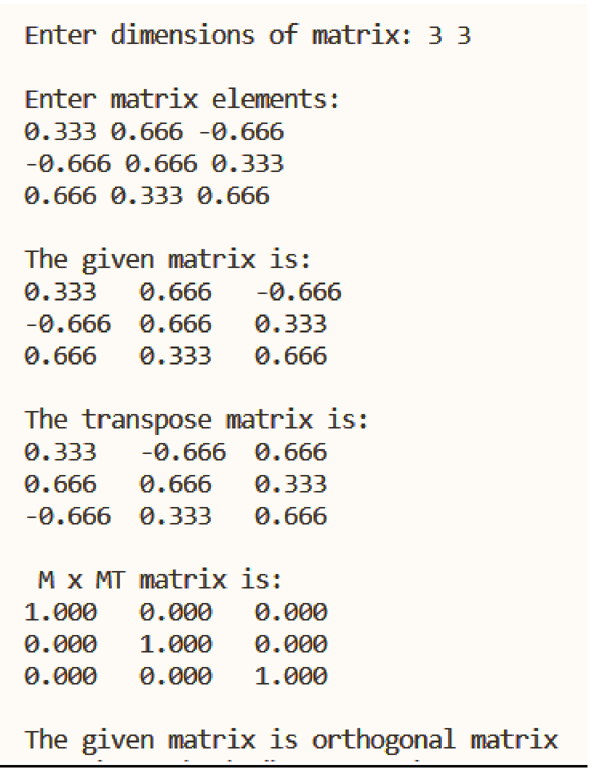
Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

float **readMatrix(int , int );
void display(float **, int , int , int );
float **transposeMatrix(float **, int , int );
float **multiplyMatrix(float **, int , int , float **, int , int );
int checkOrthogonal(float **, int , int );


int main()
{
    int n, m, p, i, j;
    float **matrix, **matT, **matXMatT;

    printf("\nEnter dimensions of matrix: ");
    scanf("%d%d", &n, &m);

    matrix = readMatrix(n, m);
    printf("\nThe given matrix is:\n");
    display(matrix, n, m, 0);

    matT = transposeMatrix(matrix, n, m);
    printf("\nThe transpose matrix is:\n");
    display(matT, m, n, 0);

    matXMatT = multiplyMatrix(matrix, n, m, matT, m, n);
    printf("\n M x MT matrix is:\n");
    display(matXMatT, n, n, 1);

    if(checkOrthogonal(matXMatT, n, n))
      printf("\nThe given matrix is orthogonal matrix");
    else
      printf("\nThe given matrix is NOT orthogonal matrix");

      return 0;
}

float **multiplyMatrix(float **matA, int r1, int c1, float **matB, int r2,
int c2) {
    if(r1 != c2)
        return NULL;

    int i, j, k;
    float **res;
    res = (float** )malloc(sizeof(float* ) * r1);
    for(i=0;i<r1;i++)
        res[i] = (float* )malloc(sizeof(float ) * c2);

    for (i = 0; i < r1; ++i) {
        for (j = 0; j < c2; ++j) {
            *(*(res + i) + j) = 0.0;
            for (k = 0; k < c1; ++k) {
                *(*(res + i) + j) += *(*(matA + i) + k) * *(*(matB + k) +
j);
            }
        }
    }
    return res;
}

float **transposeMatrix(float **matrix, int n, int m) {
    float **matrixT;
    int i, j;
    matrixT = (float** )malloc(sizeof(float* ) * n);
    for(i=0;i<m;i++)
        matrixT[i] = (float* )malloc(sizeof(float ) * m);

    for(i=0;i<n;i++) {
        for(j=0;j<m;j++){
            *(*(matrixT + j) + i) = *(*(matrix + i) + j);
        }
    }
    return matrixT;
}

int checkOrthogonal(float **matrix, int n, int m) {
      int i,j,c=0;
      for(i=0;i<n;i++) {
            for(j=0;j<m;j++) {
                  if(i == j)
                        if(roundf(*(*(matrix + i) + j)) == 1)
                              c++;
            }
      }
      return (c == n)? 1 : 0;
}

float **readMatrix(int n, int m) {
    float **matrix;
    int i, j;
    matrix = (float** )malloc(sizeof(float* ) * m);
    for(i=0;i<n;i++)
        matrix[i] = (float* )malloc(sizeof(float ) * n);
    printf("\nEnter matrix elements: \n");
    for(i=0;i<n;i++) {
        for(j=0;j<m;j++)
            scanf("%f", &*(*(matrix + i) + j));
    }

    return matrix;
}

void display(float **matrix, int n, int m, int flag) {
    int i, j;
    for(i=0;i<n;i++) {
        for(j=0;j<m;j++){
            if(flag)
                printf("%.3f\t", fabs(roundf(*(*(matrix + i) + j))));
            else
                printf("%.3f\t", *(*(matrix + i) + j));
        }
        printf("\n");
    }
}
```

Output:



```
Enter dimensions of matrix: 3 3

Enter matrix elements:
0.428 0.286 0.857
-0.857 0.428 0.286
0.286 0.857 -0.428

The given matrix is:
0.428   0.286   0.857
-0.857  0.428   0.286
0.286   0.857   -0.428

The transpose matrix is:
0.428   -0.857  0.286
0.286   0.428   0.857
0.857   0.286   -0.428

 M x MT matrix is:
1.000   0.000   0.000
0.000   1.000   0.000
0.000   0.000   1.000

The given matrix is orthogonal matrix
```



```
Enter dimensions of matrix: 3 3

Enter matrix elements:
0.333 0.666 -0.666
-0.666 0.666 0.333
0.666 0.333 0.666

The given matrix is:
0.333   0.666   -0.666
-0.666  0.666   0.333
0.666   0.333   0.666

The transpose matrix is:
0.333   -0.666  0.666
0.666   0.666   0.333
-0.666  0.333   0.666

 M x MT matrix is:
1.000   0.000   0.000
0.000   1.000   0.000
0.000   0.000   1.000

The given matrix is orthogonal matrix
```

Question 6:

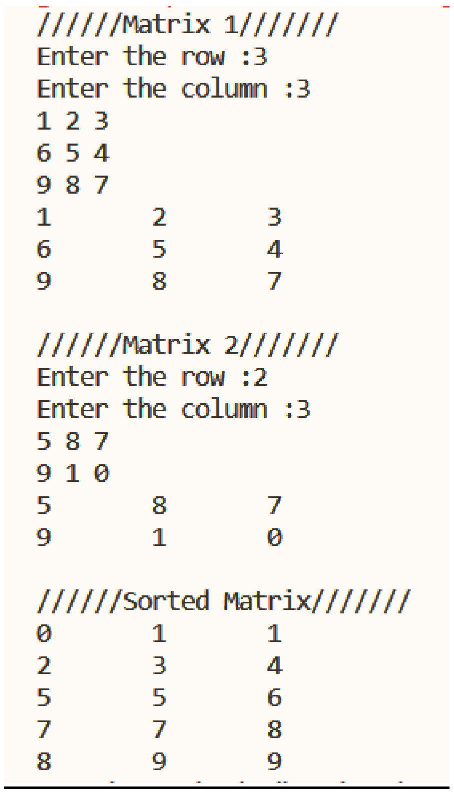
Problem Statement:

Write a function to take two 2-D arrays, sort those two arrays, then merge them into a third array that will also be sorted.

Source Code:

```
#include<stdio.h>
#include<stdlib.h>
int** createMatrix(int row,int col){
    int **resArr = (int **)malloc(sizeof(int)*row);
    if(!resArr){
        printf("Memory Error\n");
        return NULL;
    }
    for(int i=0;i<row;i++){
        resArr[i]=(int*)malloc(sizeof(int)*col);
        if(!resArr[i]){
            printf("Memory Error\n");
            return NULL;
        }
    }
    return resArr;
}
int** sort(int **arr, int row,int col){
    int *rArr = (int*)malloc(sizeof(int)*row*col);
    int k=0;
    for(int i=0;i<row;i++){
        for(int j=0;j<col;j++){
            rArr[k++]=arr[i][j];
        }
    }

    for(int i=0;i<row*col;i++){
        for(int j=i+1;j<row*col;j++){
            if(rArr[i]>rArr[j]){
                int temp=rArr[i];
                rArr[i]=rArr[j];
                rArr[j]=temp;
            }
        }
    }
    k=0;;
    for(int i=0;i<row;i++){
        for(int j=0;j<col;j++){
            arr[i][j]=rArr[k++];
        }
    }
    return arr;
}
void display(int **arr, int row, int col){
    for(int i=0;i<row;i++){
        for(int j=0;j<col;j++){
            printf("%d\t", arr[i][j]);
        }
        printf("\n");
    }
}
int** MergeMatrix(int** arr1,int** arr2, int row1,int col1,int row2,int
col2){
    int **rArr = createMatrix(row1+row2,col1);
    int i=0,j=0;
    for(i=0;i<row1;i++){
        for(j=0;j<col1;j++){
            rArr[i][j]=arr1[i][j];
        }
    }
    int k;
    for(k=0;i<row1+row2,k<row2;i++,k++){
        for(j=0;j<col1;j++){
            rArr[i][j]=arr2[k][j];
        }
    }

    return sort(rArr, row1+row2,col1);

}

int main(){
    int m,n,m1,n1;
    printf("//////Matrix 1///////\n");
    printf("Enter the row :");
    scanf("%d", &m);
    printf("Enter the column :");
    scanf("%d", &n);
    int **arr = createMatrix(m,n);
    for(int i=0;i<m;i++){
        for(int j=0;j<n;j++){
            scanf("%d", &arr[i][j]);
        }
    }
    display(arr,m,n);
    printf("\n");

    printf("//////Matrix 2///////\n");
    printf("Enter the row :");
    scanf("%d", &m1);
    printf("Enter the column :");
    scanf("%d", &n1);
    int **arr1 = createMatrix(m1,n1);
    for(int i=0;i<m1;i++){
        for(int j=0;j<n1;j++){
            scanf("%d", &arr1[i][j]);
        }
    }
    display(arr1,m1,n1);
    printf("\n");
    printf("//////Sorted Matrix///////\n");
    int **rArr = MergeMatrix(arr,arr1,m,n,m1,n1);
    display(rArr,m+m1,n);
    return 0;
}
```

Output:



```
//////Matrix 1///////
Enter the row :3
Enter the column :3
1 2 3
6 5 4
9 8 7
1       2       3
6       5       4
9       8       7

//////Matrix 2///////
Enter the row :2
Enter the column :3
5 8 7
9 1 0
5       8       7
9       1       0

//////Sorted Matrix///////
0       1       1
2       3       4
5       5       6
7       7       8
8       9       9
```

Question 9:

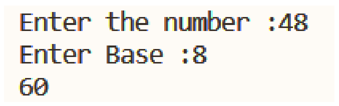
Problem Statement:

Write a function to convert a decimal number to any other base given by the user.

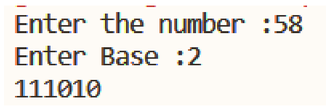
Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>


int main()
{
      char hex[] = {
            '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B',
'C', 'D', 'E', 'F'
      };

      char *ans = (char*)malloc(sizeof(char) * 100);

    float in;
    printf("Enter the number :");
      scanf("%f", &in);

      float diff = in - floor(in);
      int val = in;
      int b = 16;
    printf("Enter Base :");
      scanf("%d", &b);
      int k = 0;
      int rem;
      do {
            rem = val % b;
            ans[k++] = hex[rem];
            val = val / b;

      } while (val >= b);
      ans[k] = hex[val];

      int l = 0;
      int h = k;
      while (l < h) {
            int temp = ans[l];
            ans[l] = ans[h];
            ans[h] = temp;
            l++;
            h--;
      }
      if (diff) {
            ans[++k] = '.';
            for (int i = 1; i <= 10; i++) {
                    float temp = diff * b;
                    ans[++k] = hex[(int)temp];
                    diff = temp - floor(temp);
                    if (diff == 0) break;
                }
        }

        ans[k+1]='\0';
          printf("%s\n", ans);

          return 0;
}
```

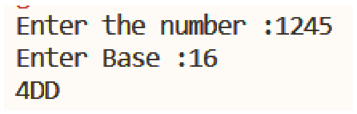
Output:



```
Enter the number :48
Enter Base :8
60
```



```
Enter the number :58
Enter Base :2
111010
```



```
Enter the number :1245
Enter Base :16
4DD
```

Question 12:

Problem Statement:

Write a program to create a linked list of n integers. Write a menu driven program to do the following

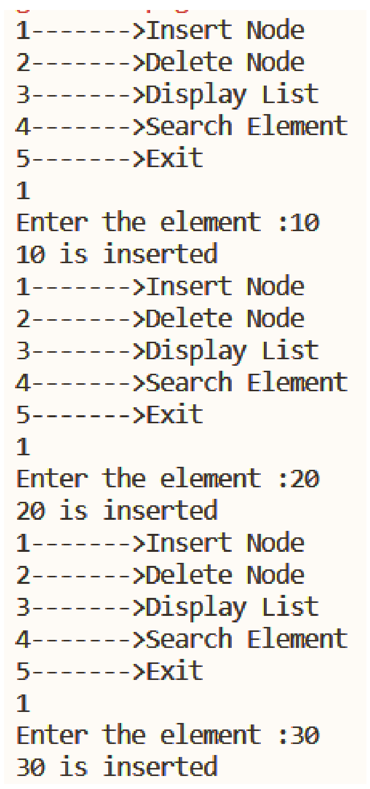
- Insert a new node
- Delete a node specified by the user
- Print the list
- Search for an element in the list.

Source Code:

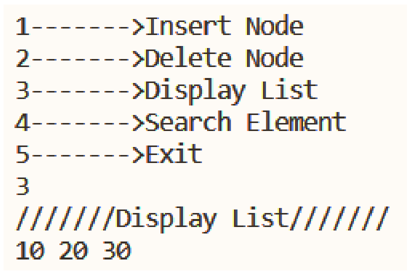
```
#include<stdio.h>
#include<stdlib.h>
struct node{
    int data;
    struct node *next;
};
struct node *head=NULL;
void insertNode(int n){
    struct node *newNode = (struct node*)malloc(sizeof(struct node));
    if(newNode){
        newNode->data = n;
        newNode->next = NULL;
        if(!head){
            head = newNode;
            printf("%d is inserted \n", head->data);
        }
        else{
            struct node *temp = head;
            while(temp->next){
                temp = temp->next;
            }
            temp->next = newNode;
            printf("%d is inserted \n", temp->next->data);
        }
    }
}
void deleteNode(int n){
    if(!head){
        printf("Node is empty\n");
    }
    else if((!head->next) && head->data==n){
        struct node *temp=head;
        head=NULL;
        printf("%d is deleted\n", temp->data);
        free(temp);
    }
    else if(head->data==n){
        struct node *temp=head;
        head= head->next;
        printf("%d is deleted\n", temp->data);
        temp=NULL;
        free(temp);
    }
    else{
        struct node *temp = head;
        struct node *temp1 = head;
        while(temp->next && temp->data!=n){
            temp1=temp;
            temp=temp->next;
        }
        if(temp->data==n){
            temp1->next=temp->next;
            temp->next = NULL;
            printf("%d is deleted\n", temp->data);
            free(temp);

        }
        else{
            printf("%d is not in the list\n", n);
        }
    }
}
```

```
void display(){
    if(!head){
        printf("List is empty\n");
    }
    else{
        struct node* temp = head;
        printf("//////Display List///////\n");
        while(temp){
            printf("%d ", temp->data);
            temp=temp->next;
        }
        printf("\n");
    }
}
int search(int n){
    if(!head){
        printf("List is empty\n");
    }
    else{
        struct node *temp = head;
        while(temp){
            if(temp->data == n){
                return 1;
            }
            temp=temp->next;
        }
        return 0;

    }
}
int main(){

    int ch;
    int n;
    while(1){
        printf("1------->Insert Node\n");
        printf("2------->Delete Node\n");
        printf("3------->Display List\n");
        printf("4------->Search Element\n");
        printf("5------->Exit\n");
        scanf("%d", &ch);
        switch(ch){
            case 1:
            printf("Enter the element :");
            scanf("%d", &n);
            insertNode(n);
            break;
            case 2:
            printf("Enter the element you want to delete :");
            scanf("%d", &n);
            deleteNode(n);
            break;
            case 3:
            display();
            break;
            case 4:
            printf("Enter the element :");
            scanf("%d", &n);
            if(search(n)){
                printf("%d is there in the list", n);
            }
            else{
                printf("%d is not there in the list", n);
            }
            break;
            case 5: exit(1);
            default: printf("Enter the right choice \n");
            break;
        }

    }

    return 0;

}
```

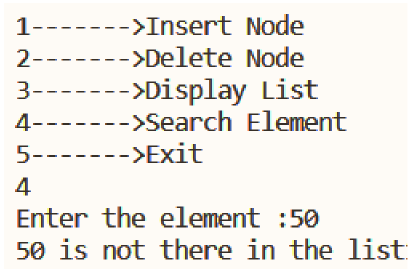
Output:



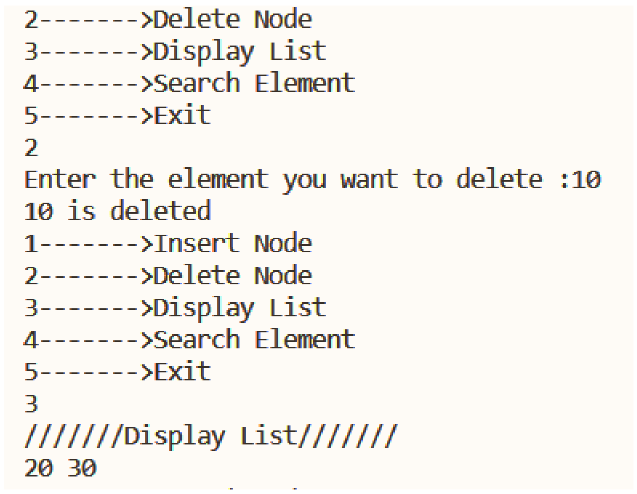
```
1------->Insert Node
2------->Delete Node
3------->Display List
4------->Search Element
5------->Exit
1
Enter the element :10
10 is inserted
1------->Insert Node
2------->Delete Node
3------->Display List
4------->Search Element
5------->Exit
1
Enter the element :20
20 is inserted
1------->Insert Node
2------->Delete Node
3------->Display List
4------->Search Element
5------->Exit
1
Enter the element :30
30 is inserted
```



```
1------->Insert Node
2------->Delete Node
3------->Display List
4------->Search Element
5------->Exit
3
///////Display List///////
10 20 30
```



```
1------->Insert Node
2------->Delete Node
3------->Display List
4------->Search Element
5------->Exit
4
Enter the element :50
50 is not there in the list
```



```
2------->Delete Node
3------->Display List
4------->Search Element
5------->Exit
2
Enter the element you want to delete :10
10 is deleted
1------->Insert Node
2------->Delete Node
3------->Display List
4------->Search Element
5------->Exit
3
///////Display List///////
20 30
```

SET-3

Question 2:

Problem Statement:

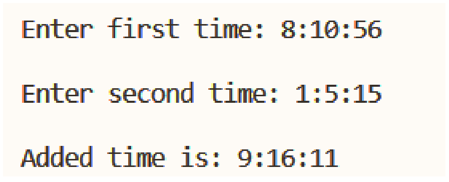
The time is specified by hours, min, sec. Write a program to add two time values given at the time of execution.

Use function for this addition and return the value to the called function.

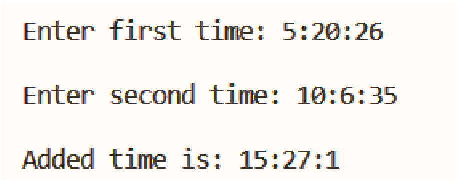
Source Code:

```
#include <stdio.h>

typedef struct {
    int hour;
    int min;
    int second;
}time;

int update(int *val) {
    if(*val > 59) {
        *val = *val - 60;
        return 1;
    }
    return 0;
}

time addTime(time a, time b) {
    time temp;
    int c = 0;
    temp.second = a.second + b.second;
    c = update(&temp.second);
    temp.min = a.min + b.min + c;
    c = update(&temp.min);
    temp.hour = a.hour + b.hour + c;
    return temp;
}

int main()
{
    time t1, t2, t3;
    printf("\nEnter first time: ");
    scanf("%d:%d:%d", &t1.hour, &t1.min, &t1.second);
    printf("\nEnter second time: ");
    scanf("%d:%d:%d", &t2.hour, &t2.min, &t2.second);
    t3 = addTime(t1, t2);
    printf("\nAdded time is: %d:%d:%d", t3.hour, t3.min, t3.second);
}
```

Output:



```
Enter first time: 8:10:56

Enter second time: 1:5:15

Added time is: 9:16:11
```



```
Enter first time: 5:20:26

Enter second time: 10:6:35

Added time is: 15:27:1
```

Question 3:

Problem Statement:

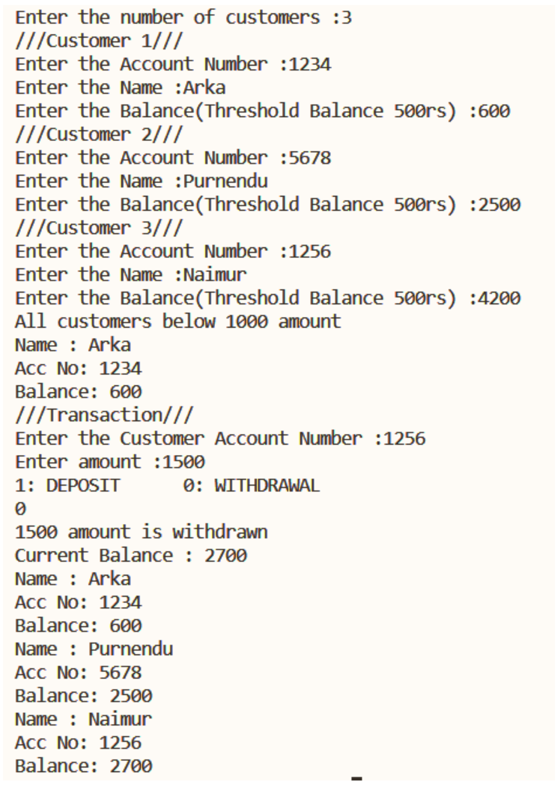
Create a structure to specify data of customers in a bank. The data to be stored is: Account number, Name, Balance in account. Assume there can have more than 100 customers in the bank.

1- Write a function to print the account number and name of each customer with balance below Rs.1000.
2- Consider that a customer request for withdrawal or deposit is given in the form: Acct.no, amount, code (1 for deposit, 0 for withdrawal)and write a program to deposit and withdraw the amount from the specified account and give a message “The balance is insufficient for the specified withdrawal” if balance is below the threshold.

Source Code:

```
#include<stdio.h>
#include<stdlib.h>
#define THRESHOLD 500
struct cust{
    int accNo;
    char name[100];
    int balance;
};
void displayBelowAmount(struct cust *bank, int n, int amount){
    printf("All customers below %d amount\n", amount);
    for(int i=0; i<n; i++){
        if(bank[i].balance < amount)
            printf("Name : %sAcc No: %d\nBalance: %d\n",
bank[i].name,bank[i].accNo,bank[i].balance);
    }
}
void display(struct cust *bank, int n){
    for(int i=0; i<n; i++){
        printf("Name : %sAcc No: %d\nBalance: %d\n",
bank[i].name,bank[i].accNo,bank[i].balance);
    }
}


void withdrawl(struct cust *bank,int amount, int custIndex){
    if(bank[custIndex].balance-amount < THRESHOLD){
        printf("Insufficient Balance(Rs %d.) to withdraw.\n", THRESHOLD);
        return;
    }
    bank[custIndex].balance -= amount;
    printf("%d amount is withdrawn \n", amount);
    printf("Current Balance : %d\n", bank[custIndex].balance);
}
void deposit(struct cust *bank,int amount, int custIndex){
    bank[custIndex].balance += amount;
    printf("%d amount is deposited \n", amount);
    printf("Current Balance : %d\n", bank[custIndex].balance);
}
int getCust(struct cust *bank, int n, int AccNo){
    for(int i=0;i<n;i++){
        if(bank[i].accNo==AccNo)
            return i;
    }
    return -1;
}
void operations(struct cust *bank,int n,int AccNo, int amount,int choice){
    int custIndex = getCust(bank,n,AccNo);
    if(choice)
        deposit(bank,amount,custIndex);
    else
        withdrawl(bank,amount,custIndex);
}
int isPresent(struct cust *bank, int n, int AccNo){
    for(int i=0;i<n;i++){
        if(bank[i].accNo==AccNo)
            return 0;
    }
    return 1;
}

int main(){
    int n;
    printf("Enter the number of customers :");
    scanf("%d",&n);
    struct cust bank[100];
    /// Insert customers ///
    for(int i=0; i<n; i++){
        printf("///Customer %d///\n", i+1);
        printf("Enter the Account Number :");
        int tempAccNo;
        scanf("%d", &tempAccNo);
        if(!isPresent(bank,i,tempAccNo)){
            printf("Account Number already exists\n");
            i--;
            continue;
        }
        printf("Enter the Name :");
        getchar();
        fgets(bank[i].name,100,stdin);
        bank[i].accNo = tempAccNo;
        printf("Enter the Balance(Threshold Balance 500rs) :");
        scanf("%d", &bank[i].balance);
    }
    displayBelowAmount(bank, n, 1000);
    printf("///Transaction///\n");
    printf("Enter the Customer Account Number :");
    int AccNo,amount;
    scanf("%d",&AccNo);
    printf("Enter amount :");
    scanf("%d",&amount);
    printf("1: DEPOSIT\t0: WITHDRAWAL\n");
    int ch;
    scanf("%d",&ch);
    operations(bank,n,AccNo,amount,ch);
    display(bank,n);

    return 0;
}
```

Output:



```
Enter the number of customers :3
///Customer 1///
Enter the Account Number :1234
Enter the Name :Arka
Enter the Balance(Threshold Balance 500rs) :600
///Customer 2///
Enter the Account Number :5678
Enter the Name :Purnendu
Enter the Balance(Threshold Balance 500rs) :2500
///Customer 3///
Enter the Account Number :1256
Enter the Name :Naimur
Enter the Balance(Threshold Balance 500rs) :4200
All customers below 1000 amount
Name : Arka
Acc No: 1234
Balance: 600
///Transaction///
Enter the Customer Account Number :1256
Enter amount :1500
1: DEPOSIT      0: WITHDRAWAL
0
1500 amount is withdrawn
Current Balance : 2700
Name : Arka
Acc No: 1234
Balance: 600
Name : Purnendu
Acc No: 5678
Balance: 2500
Name : Naimur
Acc No: 1256
Balance: 2700
```

Question 6:

Problem Statement:

Consider that a large binary matrix is stored in a file. Each line is a row of the matrix. The dimensions of the matrix are not known in advance. Write a program to read the matrix into a dynamic array, find its dimension, computer row-sums and create a new file to store row-no and the corresponding row sum.

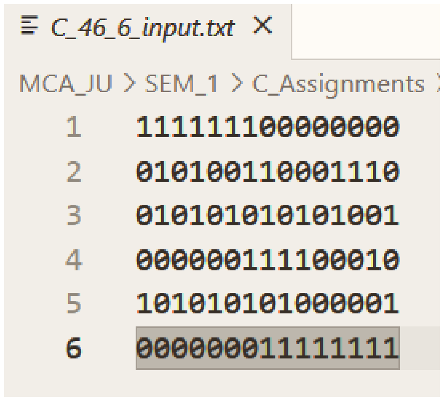
Source Code:

```
#include <stdio.h>
#include <ctype.h>
#include <malloc.h>

int** getMatrix(int **, int *, int *);
int** findRowSum(int **, int , int );
void displayMatrix(int **, int , int );

int main()
{
    int rowCount = 0, colCount = 0;
    int **matrix, **rowSum;

    matrix = getMatrix(matrix, &rowCount, &colCount);
    printf("\nThe dimensions of the matrix is: %d X %d", rowCount,
colCount);
    //displayMatrix(matrix, rowCount, colCount);
    printf("\nThe row sum of the matrix is:");
    rowSum = findRowSum(matrix, rowCount, colCount);
    displayMatrix(rowSum, rowCount , 2);

    return 0;
}

int** findRowSum(int **matrix, int row, int col)
{
    int **rowSum;
    int i,j;
    rowSum = (int** )malloc(sizeof(int* ) * row);
    for(i=0;i<row;i++)
        rowSum[i] = (int* )malloc(sizeof(int ) * 2);

    for(i=0;i<row;i++){
        *(*(rowSum + i) + 0) = i;
        int sum = 0;
        for(j=0;j<col;j++)
            sum += *(*(matrix + i) + j);
        *(*(rowSum + i) + 1) = sum;
    }

    return rowSum;
```

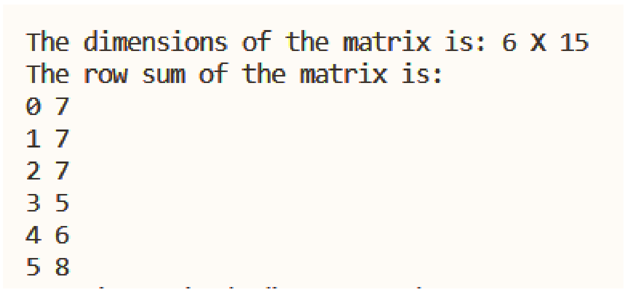
```
}

int** getMatrix(int **matrix, int *rowCount, int *colCount)
{
    char c;
    int i,j;
    FILE *in = fopen("C_46_6_input.txt", "r");
    while(1) {
        fflush(stdin);
        c = fgetc(in);
        if(c == EOF)
            break;
        else if(c == '\n') {
            *rowCount = *rowCount + 1;
            *colCount = 0;
        }
        else
            *colCount = *colCount + 1;
    }
    fclose(in);
    *rowCount = *rowCount + 1;

    matrix = (int** )malloc(sizeof(int* ) * (*rowCount));
    for(i=0;i<*rowCount;i++)
        matrix[i] = (int *)malloc(sizeof(int ) * (*colCount));


    in = fopen("C_46_6_input.txt", "r");
    i = 0;
    j = -1;
    while(1) {
        fflush(stdin);
        c = fgetc(in);
        if(c == EOF)
            break;
        else if(c == '\n') {
            i++;
            j=-1;
        }
        else {
            j++;
            *(*(matrix + i) + j) = c-48;
        }
    }
    fclose(in);
    return matrix;
}

void displayMatrix(int **matrix, int row, int col)
{
    printf("\n");
    int i,j;
    for(i=0;i<row;i++){
        for(j=0;j<col;j++)
```

```
            printf("%d ", *(*(matrix + i) + j));
        printf("\n");
    }
}
```

Output:



```
C_46_6_input.txt ×
MCA_JU > SEM_1 > C_Assignments
1   111111100000000
2   010100110001110
3   010101010101001
4   000000111100010
5   101010101000001
6   000000011111111
```



```
The dimensions of the matrix is: 6 X 15
The row sum of the matrix is:
0 7
1 7
2 7
3 5
4 6
5 8
```

SET-4

Question 5:

Problem Statement:

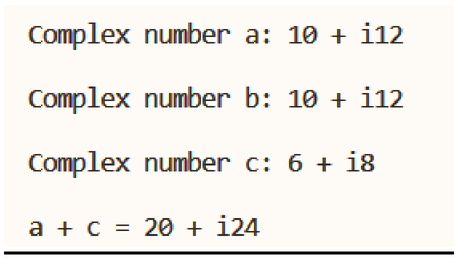
Write a C++ program to create a class complex having two integers real and imaginary. Create a three constructors function taking no argument, one argument and two arguments for three constructors. Show () and sum() functions are member functions, displaying and finding the addition of two objects respectively.

Source Code:

```
#include <iostream>

class Complex {
private:
    int real, img;
public:
    Complex() {};
    Complex(const Complex& object) {
        this->real = object.real;
        this->img = object.img;
    }
    Complex(const int& real, const int& img) {
        this->real = real;
        this->img = img;
    }

    void show() {
        std::cout<<this->real<<" + i"<<this->img<<"\n";
    }

    Complex sum(const Complex& ob) {
        Complex temp;
        temp.real = this->real + ob.real;
        temp.img = this->img + ob.img;
        return temp;
    }

    Complex operator+(const Complex ob) {
        Complex temp;
        temp.real = this->real + ob.real;
        temp.img = this->img + ob.img;
        return temp;
    }

};
```

```
int main()
{
    Complex *a = new Complex(10, 12);
    Complex *b = a;
    Complex *c = new Complex(6, 8);
    std::cout<<"\nComplex number a: ";
    a->show();
    std::cout<<"\nComplex number b: ";
    b->show();
    std::cout<<"\nComplex number c: ";
    c->show();

    Complex x = a->sum(*b);
    //*b = *a + *c;
    std::cout<<"\na + c = ";
    x.show();
}
```

Outupt:



```
Complex number a: 10 + i12

Complex number b: 10 + i12

Complex number c: 6 + i8

a + c = 20 + i24
```

Question 7:

Problem Statement:

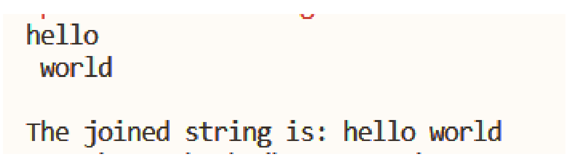
Write a C++ program to create a class string, which stores string with constructor, displays the string and joins two strings with join user defined function taking two arguments of string object.

Source Code:

```
#include <iostream>
#include <cstring>

class String {
```

```
private:
    char *str;
public:
    String() {}
    String(const char* s) {
        str = new char[strlen(s)];
        strcpy(str, s);
    }
    void show() {
        std::cout<<str<<"\n";
    }
    void join(String& s1, String& s2) {
        str = new char[(strlen(s1.str) + strlen(s2.str))];
        strcpy(str, s1.str);
        strcat(str, s2.str);
    }
};

int main()
{
    String t1("hello");
    t1.show();
    String t2(" world");
    t2.show();

    String t3;
    t3.join(t1, t2);
    std::cout<<"\nThe joined string is: ";
    t3.show();

}
```

<u>Output:</u>



```
hello
 world

The joined string is: hello world
```

Question 8:

Problem Statement:

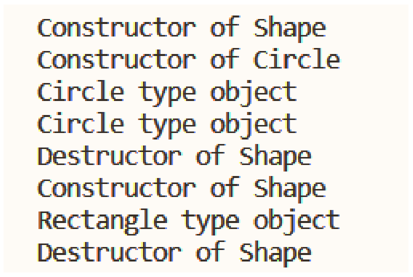
Write a C++ program to demonstrate

(A) Copy Constructor (B) Parameterized Constructor (C) Virtual destructor

Source Code:

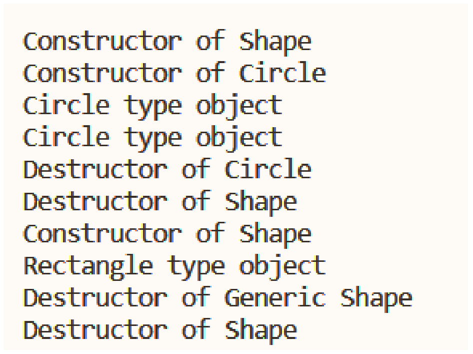
```
#include <iostream>
#include <string>

class Shape {
public:
    Shape() { std::cout<<"\nConstructor of Shape"; }
    virtual ~Shape() { std::cout<<"\nDestructor of Shape"; }
};

class Cirlce : public Shape {
private:
    std::string type;
public:
    Cirlce() {
        type = "Circle";
        std::cout<<"\nConstructor of Circle";
    }
    virtual ~Cirlce() { std::cout<<"\nDestructor of Circle"; }
    Cirlce(const Cirlce& obj) { this->type = obj.type; }
    void showType() { std::cout<<"\n"<<type<<" type object"; }
};

class GenericShape : public Shape {
private:
    std::string type;
public:
    GenericShape() { }
    GenericShape(std::string type) { this->type = type; }
    ~GenericShape() { std::cout<<"\nDestructor of Generic Shape"; }
    void showType() { std::cout<<"\n"<<type<<" type object"; }
};


int main()
{
    Cirlce *c1 = new Cirlce();
    Cirlce *c2 = c1;
    c1->showType();
    c2->showType();
    Shape *s1 = c1;

    delete s1;
    GenericShape *g1 = new GenericShape("Rectangle");
    Shape *s2 = g1;
    g1->showType();
    delete s2;
}
```

Output:



```
Constructor of Shape
Constructor of Circle
Circle type object
Circle type object
Destructor of Shape
Constructor of Shape
Rectangle type object
Destructor of Shape
```

Figure 1 Without Virtual destructor



```
Constructor of Shape
Constructor of Circle
Circle type object
Circle type object
Destructor of Circle
Destructor of Shape
Constructor of Shape
Rectangle type object
Destructor of Generic Shape
Destructor of Shape
```

Figure 2 With virtual Destructor

Question 13:

Problem Statement:

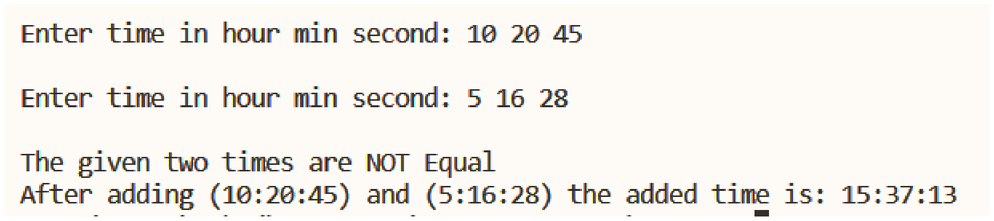
Write a C++ program to overload the following operators

- '>>' to accept time from user (in hours: mins:sec)
- '+' to add two different user-given time.
- '<<' to display the time in hours: mins: sec format.
- '==' to check whether two user-given times are equal or not.

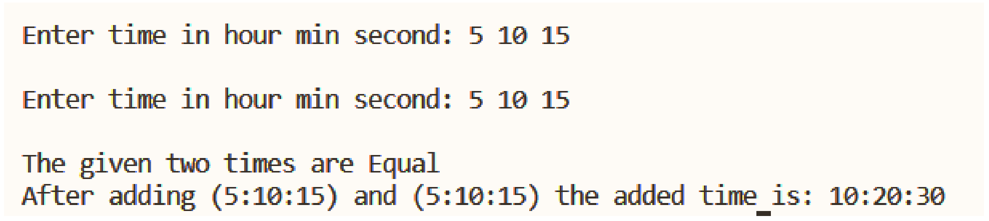
Source Code:

```
#include <iostream>

class TIME {
private:
    int hour, min, sec;

    int update(int *val) {
        if(*val > 59) {
            *val = *val - 60;
            return 1;
        }
        return 0;
    }

public:
    TIME() {}
    TIME(int hour, int min, int sec) {
        this->hour = hour;
        this->min = min;
        this->sec = sec;
    }

    friend std::ostream& operator << (std::ostream& out, const TIME& time)
{
        out << time.hour << ":" << time.min << ":" << time.sec;
        return out;
    }

    friend std::istream& operator >> (std::istream& in, TIME& time) {
        std::cout<<"\nEnter time in hour min second: ";
        in >> time.hour;
        in >> time.min;
        in >> time.sec;
        return in;
    }

    TIME operator + (const TIME& t) {
        TIME temp;
        int c = 0;
        temp.sec = this->sec + t.sec;
        c = update(&temp.sec);
        temp.min = this->min + t.min + c;
        c = update(&temp.min);
        temp.hour = this->hour + t.hour + c;
        return temp;
    }

    bool operator == (const TIME& t) {
        if(this->hour == t.hour &&
            this->min == t.min &&
            this->sec == t.sec)
            return true;
        return false;
    }

};



int main()
{
    TIME t1, t2;
    std::cin >> t1 >> t2;
    std::cout << "\nThe given two times are " << ((t1 == t2) ? "" : "NOT
") << "Equal";
    TIME t3 = t1 + t2;
    std::cout << "\nAfter adding (" << t1 << ") and (" << t2 << ") the
added time is: " << t3;

}
```

<u>Output:</u>



```
Enter time in hour min second: 10 20 45

Enter time in hour min second: 5 16 28

The given two times are NOT Equal
After adding (10:20:45) and (5:16:28) the added time is: 15:37:13
```



```
Enter time in hour min second: 5 10 15

Enter time in hour min second: 5 10 15

The given two times are Equal
After adding (5:10:15) and (5:10:15) the added time is: 10:20:30
```