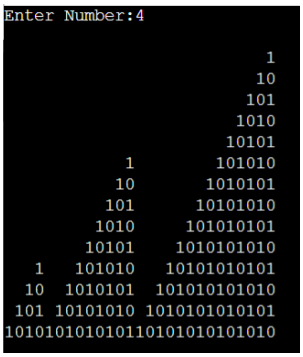
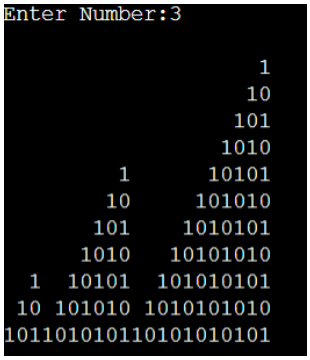
Question 1: Write a C program to print the following pattern for given input from user. [Marks: 20]





Solution:

```c
#include <stdio.h>
 int main(){
    int n; int flag=0;

    printf("Enter Number:");
    scanf("%d",&n);

     n++;
    for(int i = 1; i <= 3*n ; i++){
        flag=0;
        if(i <= 2*n)

            for(int j = 1 ; j <= (2 * n - 1)/2 ; j++)
               { printf(" ");
                }
        else{
            for(int j = 1 ; j <= (3 * n - i) ; j++)
                printf(" ");
            for(int j = 1 ; j <= (2 * (i - 2 * n) - 1)/2 ; j++)
                if(flag==0)
                {
                printf("1");
                flag=1;
                }
                else
                {
                printf("0");
                flag=0;
                }
        }

        if(i <= n){

            for(int j = 1 ; j <= (4 * n - 1)/2 ; j++)
               printf(" ");
        }
        else{ flag=0;
            for(int j = 1 ; j <= 3 * n - i ; j++)
                printf(" ");
            for(int j = 1 ; j <= (2 * (i - n) - 1)/2; j++)
               if(flag==0)
                {
                printf("1");
                flag=1;
                }
                else
                {
                printf("0");
                flag=0;
                }
         }

        for(int j = 1 ; j <= 3 * n - i ; j++)
            printf(" ");
            flag=0;
        for(int j = 1 ; j <= (2 * i - 1)/2 ; j++)
            if(flag==0)
                {
                printf("1");
                flag=1;
                }
                else
                {
                printf("0");
                flag=0;
                }

        printf("\n");
    }
}
```

Question 2: Write a C program in which a function accepts two positive integers ‘r’ and ‘unit’ and a positive integer array ‘arr’ of size ‘n’ as its argument ‘r’ represents the number of rodents present in a particular area, ‘unit’ is the amount of food each rodent consumes and each ith element of array ‘arr’ represents the amount of food present in ‘i+1’ house number, where 0 <= i. Find the number of houses in which the amount of food is sufficient for rodents.

- Return -1 if the array is null
- Return 0 if the total amount of food from all houses is not sufficient for all the rats.
- Computed values lie within the integer range.

[Marks: 15]

Input:

r: 7

unit: 2

n: 8

arr: 2 8 3 5 7 4 1 2

Output:

4

```c
#include <stdio.h>
int calculate (int r, int unit, int arr[], int n)
{
    if (n == 0)
        return -1;
    int totalFoodRequired = r * unit;
    int foodTillNow = 0;
    int house = 0;
  for (house = 0; house < n; ++house)
    {
        foodTillNow += arr[house];
        if (foodTillNow >= totalFoodRequired)
        {
            break;
        }
    }
    if (totalFoodRequired > foodTillNow)
        return 0;
    return house + 1;
}
int main ()
{
    int r;
    printf("Enter no. of rodents= ");
    scanf("%d",&r);
    int unit;
    printf("Enter no. of units each rodent consumes= ");
    scanf("%d",&unit);
    int n;
    printf("Enter no. of houses= ");
    scanf("%d",&n);
    int arr[n];
    printf("Enter no. of food units present in each house= ");
    for (int i = 0; i < n; ++i)
    {
        scanf("%d",&arr[i]);
    }
    printf("amount of food in 1st %d houses is sufficient for all the rats",calculate (r, unit, arr, n));
    return 0;
}
```

Question 3: Write the functions for checking similarity of two triangles. the structures and main is given to you with the declaration of the functions. Write the functions or any more function you need. For finding angle between three coordinates the formula is:

$\arccos((P_{12}^2 + P_{13}^2 - P_{23}^2) / (2 * P_{12} * P_{13}))$

where P_{12} is the length of the segment from P1 to P2, calculated by

$\text{sqrt}((P1_x - P2_x)^2 + (P1_y - P2_y)^2)$

[Marks: 20]

```
#include <stdio.h>
#include <math.h>
#define PI 3.1415926

struct point{
   int x;
   int y;
};

struct triangle{
   struct point point1;
   struct point point2;
   struct point point3;
};

int isSimilar(struct triangle* t1, struct triangle* t2);

int main()
{
   struct triangle t1,t2;
   printf("Enter coordinates of triangle 1 as x1,y1,x2,y2,x3,y3: ");
   scanf("%d %d %d %d %d
%d",&(t1.point1.x),&(t1.point1.y),&(t1.point2.x),&(t1.point2.y),&(t1.point3.x),&(t1.point3.y));
   printf("Enter coordinates of triangle 1 as x1,y1,x2,y2,x3,y3: ");
   scanf("%d %d %d %d %d
%d",&(t2.point1.x),&(t2.point1.y),&(t2.point2.x),&(t2.point2.y),&(t2.point3.x),&(t2.point3.y));

   if(isSimilar(&t1,&t2))
      printf("\n\nThe two triangles are similar\n");
   else
      printf("\n\nThe two triangles are NOT similar\n");

   return 0;
}
```

Solution:

```
#include <stdio.h>
#include <math.h>
#define PI 3.1415926

struct point{
    int x;
    int y;
};

struct triangle{
    struct point point1;
    struct point point2;
    struct point point3;
};

double eucledian(int x1, int y1, int x2, int y2){
    return sqrt(((x1-x2)*(x1-x2)) + ((y1-y2)*(y1-y2)));
}

double angle(struct point* p1, struct point* p2, struct point* p3){
    double P12 = eucledian(p1->x,p1->y,p2->x,p2->y);
    double P13 = eucledian(p1->x,p1->y,p3->x,p3->y);
    double P23 = eucledian(p2->x,p2->y,p3->x,p3->y);

    double angle = acos((P12*P12 + P23*P23 - P13*P13) / (2 * P12 * P23));

    return (angle/PI)*180;
}

int same2of3(double a1, double b1, double c1, double a2, double b2, double c2){
    int same=0;
    if(a1==a2){
        if((b1==b2)||(b1==c2))
            same=1;
        else if((c1==b2)||(c1==c2))
            same=1;
    }
    else if(a1==b2){
        if((b1==a2)||(b1==c2))
            same=1;
        else if((c1==a2)||(c1==c2))
            same=1;
    }
    else if(a1==c2){
        if((b1==a2)||(b1==b2))
            same=1;
        else if((c1==a2)||(c1==b2))
            same=1;
    }

    return same;
}


int isSimilar(struct triangle* t1, struct triangle* t2){
```

```
    int similar=0;
    double t1angle123, t1angle231, t1angle213, t2angle123, t2angle231, t2angle213;

    t1angle123 = angle(&(t1->point1),&(t1->point2),&(t1->point3));
    t1angle231 = angle(&(t1->point2),&(t1->point3),&(t1->point1));
    t1angle213 = angle(&(t1->point2),&(t1->point1),&(t1->point3));
    t2angle123 = angle(&(t2->point1),&(t2->point2),&(t2->point3));
    t2angle231 = angle(&(t2->point2),&(t2->point3),&(t2->point1));
    t2angle213 = angle(&(t2->point2),&(t2->point1),&(t2->point3));

    //printf("\nThe angles are %f %f %f and %f %f
%f\n",t1angle123,t1angle231,t1angle213,t2angle123,t2angle231,t2angle213);

    return same2of3(t1angle123,t1angle231,t1angle213,t2angle123,t2angle231,t2angle213);
}


int main()
{
    struct triangle t1,t2;
    printf("Enter coordinates of triangle 1 as x1,y1,x2,y2,x3,y3: ");
    scanf("%d %d %d %d %d
%d",&(t1.point1.x),&(t1.point1.y),&(t1.point2.x),&(t1.point2.y),&(t1.point3.x),&(t1.point3.y));
    printf("Enter coordinates of triangle 1 as x1,y1,x2,y2,x3,y3: ");
    scanf("%d %d %d %d %d
%d",&(t2.point1.x),&(t2.point1.y),&(t2.point2.x),&(t2.point2.y),&(t2.point3.x),&(t2.point3.y));

    if(isSimilar(&t1,&t2))
        printf("\n\nThe two triangles are similar\n");
    else
        printf("\n\nThe two triangles are NOT similar\n");


    return 0;
}
```

Question 4: Write a Program to remove all the duplicate word in this String.

[Marks: 15]

Solution:

```c
#include <stdio.h>
#include <stdlib.h>
    =#include <string.h>
     int main ()
      {   char str[100], word[100], twoD[10][30];
    int i = 0, j = 0, k = 0, len1 = 0, len2 = 0, l = 0;

    printf ("Enter the string\n");
       gets (str);

    // let us convert the string into 2D array
    for (i = 0; str[i] != '\0'; i++)
        {
        if (str[i] == ' ')
        {
            twoD[k][j] = '\0';
            k ++;
            j = 0;
        }
        else
        {
            twoD[k][j] = str[i];
            j ++;
        }
    }
        twoD[k][j] = '\0';

    j = 0;
    for (i = 0; i < k; i++)
    {
        int present = 0;
        for (l = 1; l < k + 1; l++)
        {
            if (twoD[l][j] == '\0' || l == i)
            {
                continue;
            }

            if (strcmp (twoD[i], twoD[l]) == 0) {
                twoD[l][j] = '\0';
                present = present + 1;
            }
        }
    }
    j = 0;

    for (i = 0; i < k + 1; i++)
    {
        if (twoD[i][j] == '\0')
            continue;
        else
            printf ("%s ", twoD[i]);
    }

    printf ("\n");

    return 0;
}
```

Question 5: You are given two arrays each of size n, a and b consisting of the first n positive integers each exactly once, that is, they are permutations.

Your task is to find the minimum time required to make both the arrays empty. The following two types of operations can be performed any number of times each taking 1 second:

In the first operation, you are allowed to rotate the first array clockwise.
In the second operation, when the first element of both the arrays is the same, they are removed from both the arrays and the process continues.

[Marks: 15]

Input format

The first line contains an integer n, denoting the size of the array.
The second line contains the elements of array a.
The third line contains the elements of array b.

Output format

Print the total time taken required to empty both the array.
1<=n<=100

Test case:
input: 3
1 3 2
2 3 1
output: 6

input: 5

1 2 3 4 5

2 3 5 4 1

output: 8

Solution:

```c
#include <stdio.h>

int main(){
    int n;
    scanf("%d", &n);
    int a[100],b[100],i,j;

    for (i=0;i<n;i++){
        scanf("%d",&a[i]);
    }

    for (i=0;i<n;i++){
        scanf("%d",&b[i]);
    }

    int time=0,k;
    for(k=0;k<n;){
        if(a[k]==b[k]){
            k++;
            time++;
        }
        else{
            int temp=a[k],t=k;
            for(;t<n-1;t++){
                a[t]=a[t+1];
            }
            a[t]=temp;
            time++;
        }
    }
    printf("%d",time);
}
```

Question 6: A terrorist group plans to execute people using the Josephus Problem. The problem states that there are people standing in a circle waiting to be executed. The counting out begins at some point in the circle and proceeds around the circle in a fixed direction. In each step, a certain number of people are skipped and the next person is executed. The elimination proceeds around the circle (which is becoming smaller and smaller as the executed people are removed), until only the last person remains, who is given freedom.

Write a C program to implement the Josephus problem. Use dynamic memory allocation technique. [Marks: 20]

Desired input/output pattern:

```
Enter a number: 1
Do you want to add a number [1/0]? 1
Enter a number: 2
Do you want to add a number [1/0]? 1
Enter a number: 3
Do you want to add a number [1/0]? 1
Enter a number: 4
Do you want to add a number [1/0]? 1
Enter a number: 5
Do you want to add a number [1/0]? 1
Enter a number: 6
Do you want to add a number [1/0]? 1
Enter a number: 7
Do you want to add a number [1/0]? 0
The persons are:
1   2   3   4   5   6   7
Enter the number of persons to be skipped: 3
3 has been killed.
6 has been killed.
2 has been killed.
7 has been killed.
5 has been killed.
1 has been killed.
The person to survive is : 4
```

Solution:

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

struct node
{
    int num;
    struct node *next;
};

void create(struct node **);
void display(struct node *);
int survivor(struct node **, int);

int main()
{
    struct node *head = NULL;
    int survive, skip;

    create(&head);
    printf("The persons are:\n");
    display(head);
    printf("Enter the number of persons to be skipped: ");
    scanf("%d", &skip);
    survive = survivor(&head, skip);
    printf("The person to survive is : %d\n", survive);
    free(head);

    return 0;
}

int survivor(struct node **head, int k)
{
    struct node *p, *q;
    int i;

    q = p = *head;
    while (p->next != p)
    {
        for (i = 0; i < k - 1; i++)
        {
            q = p;
            p = p->next;
        }
        q->next = p->next;
        printf("%d has been killed.\n", p->num);
        free(p);
        p = q->next;
    }
    *head = p;

    return (p->num);
}
```

```c
void create (struct node **head)
{
    struct node *temp, *rear;
    int a, ch;

    do
    {
        printf("Enter a number: ");
        scanf("%d", &a);
        temp = (struct node *)malloc(sizeof(struct node));
        temp->num = a;
        temp->next = NULL;
        if (*head == NULL)
        {
            *head = temp;
        }
        else
        {
            rear->next = temp;
        }
        rear = temp;
        printf("Do you want to add a number [1/0]? ");
        scanf("%d", &ch);
    } while (ch != 0);
    rear->next = *head;
}

void display(struct node *head)
{
    struct node *temp;

    temp = head;
    printf("%d   ", temp->num);
    temp = temp->next;
    while (head != temp)
    {
        printf("%d   ", temp->num);
        temp = temp->next;
    }
    printf("\n");
}
```