

PRACTICE QUESTIONS

1) Storing and printing an array:

- Code:

```
#include <stdio.h>

void main()
{
    int arr[10];
    int i;
    printf("\n\nRead and Print elements of an
array:\n");
    printf("-----\n");

    printf("Input 10 elements in the array :\n");
    for(i=0; i<10; i++)
    {
        printf("element - %d : ",i);
        scanf("%d", &arr[i]);
    }

    printf("\nElements in array are: ");
    for(i=0; i<10; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");}
```

2) Store and print the element of the array in reverse order

- Code:

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

```
void main()
```

```
{
```

```
    int i,n,a[100];
```

```
        printf("\n\nRead n number of values in an  
array and display it in reverse order:\n");
```

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

```
        printf("Input the number of elements to store in  
the array :");
```

```
        scanf("%d",&n);
```

```
        printf("Input %d number of elements in the  
array :\n",n);
```

```
        for(i=0;i<n;i++)
```

```
        {
```

```
            printf("element - %d : ",i);
```

```
            scanf("%d",&a[i]);
```

```
        }
```

```
        printf("\nThe values store into the array are :  
\n");
```

```
        for(i=0;i<n;i++)
```

```

        {
            printf("% 5d",a[i]);
        }

        printf("\n\nThe values store into the array in
reverse are :\n");
        for(i=n-1;i>=0;i--)
        {
            printf("% 5d",a[i]);
        }
        printf("\n\n");
    }

```

3) Sorting the elements of the array

- Code:

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

```
void main()
```

```
{
    int arr1[100];
    int n, i, j, tmp;
```

```

        printf("\n\nsort elements of array in ascending
order :\n ");
        printf("-----
\n");

```

```

printf("Input the size of array : ");
scanf("%d", &n);

printf("Input %d elements in the array :\n",n);
for(i=0;i<n;i++)
{
    printf("element - %d : ",i);
    scanf("%d",&arr1[i]);
}

for(i=0; i<n; i++)
{
    for(j=i+1; j<n; j++)
    {
        if(arr1[j] <arr1[i])
        {
            tmp = arr1[i];
            arr1[i] = arr1[j];
            arr1[j] = tmp;
        }
    }
}

printf("\nElements of array in sorted ascending
order:\n");
for(i=0; i<n; i++)
{
    printf("%d ", arr1[i]);
}

```

```

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

```

4) Finding maximum and minimum elements in an array

- Code:

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

```
void main()
```

```
{
```

```
    int arr1[100];
```

```
    int i, mx, mn, n;
```

```

        printf("\n\nFind maximum and minimum
        element in an array :\n");

```

```

        printf("-----\n");

```

```

        printf("Input the number of elements to be
        stored in the array :");

```

```
        scanf("%d",&n);
```

```

        printf("Input %d elements in the array
        :\n",n);

```

```
        for(i=0;i<n;i++)
```

```
        {
```

```

        printf("element - %d : ",i);
        scanf("%d",&arr1[i]);
    }

    mx = arr1[0];
    mn = arr1[0];

    for(i=1; i<n; i++)
    {
        if(arr1[i]>mx)
        {
            mx = arr1[i];
        }

        if(arr1[i]<mn)
        {
            mn = arr1[i];
        }
    }
    printf("Maximum element is : %d\n", mx);
    printf("Minimum element is : %d\n\n", mn);
}

```

5) Frequency of each element in an array

- Code:

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

```

void main()
{
    int arr1[100], fr1[100];
    int n, i, j, ctr;

    printf("\n\nCount frequency of each element
of an array:\n");
    printf("-----
---\n");

    printf("Input the number of elements to be
stored in the array :");
    scanf("%d",&n);

    printf("Input %d elements in the array
:\n",n);
    for(i=0;i<n;i++)
    {
        printf("element - %d : ",i);
        scanf("%d",&arr1[i]);
        fr1[i] = -1;
    }
    for(i=0; i<n; i++)
    {
        ctr = 1;
        for(j=i+1; j<n; j++)
        {

```

```

        if(arr1[i]==arr1[j])
        {
            ctr++;
            fr1[j] = 0;
        }
    }

    if(fr1[i]!=0)
    {
        fr1[i] = ctr;
    }
}

printf("\nThe frequency of all elements of
array : \n");
for(i=0; i<n; i++)
{
    if(fr1[i]!=0)
    {
        printf("%d occurs %d times\n", arr1[i],
fr1[i]);
    }
}
}

```

✓ 6) Delete an element from a desired position

- Code:

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

```
void main(){
```



```

int arr1[50],i,pos,n;

    printf("\n\nDelete an element at desired
position from an array :\n");
    printf("-----\n");

    printf("Input the size of array : ");
    scanf("%d", &n);
    /* Stored values into the array*/
    printf("Input %d elements in the array in
ascending order:\n",n);
    for(i=0;i<n;i++)
    {
        printf("element - %d : ",i);
        scanf("%d",&arr1[i]);
    }

    printf("\nInput the position where to delete: ");
    scanf("%d",&pos);
    /*---- locate the position of i in the array -----*/
    i=0;
    while(i!=pos-1)
        i++;
    /*---- the position of i in the array will be
replaced by the
value of its right */
    while(i<n){

```

```

        arr1[i]=arr1[i+1];
        i++;}

n--;
printf("\nThe new list is : ");
for(i=0;i<n;i++)
{
    printf(" %d",arr1[i]);}
printf("\n\n");}

```

✓ 7) Storing the 2d-array and printing the matrix

- Code:

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

```
void main()
```

```
{
    int arr1[3][3],i,j;
```

```
    printf("\n\nRead a 2D array of size 3x3 and
    print the matrix :\n");
```

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

```
    /* Stored values into the array*/
```

```
    printf("Input elements in the matrix :\n");
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
```

```

        printf("element - [%d],[%d] : ",i,j);
        scanf("%d",&arr1[i][j]);
    }
}

printf("\nThe matrix is : \n");
for(i=0;i<3;i++)
{
    printf("\n");
    for(j=0;j<3;j++)
        printf("%d\t",arr1[i][j]);
}
printf("\n\n");
}

```

✓ 8) Addition of matrix of same dimension

- Code:

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

```
void main()
```

```
{
```

```
int arr1[50][50],brr1[50][50],crr1[50][50],i,j,n;
```

```
printf("\n\nAddition of two Matrices :\n");
```

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

```
printf("Input the size of the square matrix  
(less than 5): ");
```

```
scanf("%d", &n);
```

```

/* Stored values into the array*/
printf("Input elements in the first matrix
:\n");
for(i=0;i<n;i++)
{
    for(j=0;j<n;j++)
    {
        printf("element - [%d],[%d] : ",i,j);
        scanf("%d",&arr1[i][j]);
    }
}

printf("Input elements in the second matrix
:\n");
for(i=0;i<n;i++)
{
    for(j=0;j<n;j++)
    {
        printf("element - [%d],[%d] : ",i,j);
        scanf("%d",&brr1[i][j]);
    }
}

printf("\nThe First matrix is :\n");
for(i=0;i<n;i++)
{
    printf("\n");
    for(j=0;j<n;j++)
        printf("%d\t",arr1[i][j]);
}

```

```

    }

    printf("\nThe Second matrix is :\n");
    for(i=0;i<n;i++)
    {
        printf("\n");
        for(j=0;j<n;j++)
            printf("%d\t",brr1[i][j]);
    }
    /* calculate the sum of the matrix */
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            crr1[i][j]=arr1[i][j]+brr1[i][j];
    printf("\nThe Addition of two matrix is : \n");
    for(i=0;i<n;i++){
        printf("\n");
        for(j=0;j<n;j++)
            printf("%d\t",crr1[i][j]);
    }
    printf("\n\n");
}

```

✓ 9) Multiplication of matrices

- Code:

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

```
void main()
```

```
{
```

```

int
arr1[50][50],brr1[50][50],crr1[50][50],i,j,k,r1,c1,
r2,c2,sum=0;

```

```

printf("\n\nMultiplication of two Matrices
:\n");
printf("-----\n");

```

```

printf("\nInput the rows and columns of first
matrix : ");
scanf("%d %d",&r1,&c1);
printf("\nInput the rows and columns of second
matrix : ");

```

```

scanf("%d %d",&r2,&c2);
if(c1!=r2){
printf("Mutiplication of Matrix is not
possible.");
printf("\nColumn of first matrix and row of
second matrix must be same.");
}

```

```

else
{
printf("Input elements in the first matrix
:\n");
for(i=0;i<r1;i++)
{
for(j=0;j<c1;j++)
{

```

```

        printf("element - [%d],[%d] : ",i,j);
        scanf("%d",&arr1[i][j]);
    }
}
printf("Input elements in the second matrix
:\n");
for(i=0;i<r2;i++)
{
    for(j=0;j<c2;j++)
    {
        printf("element - [%d],[%d] : ",i,j);
        scanf("%d",&brr1[i][j]);
    }
}
printf("\nThe First matrix is :\n");
    for(i=0;i<r1;i++)
    {
        printf("\n");
        for(j=0;j<c1;j++)
        printf("%d\t",arr1[i][j]);
    }

printf("\nThe Second matrix is :\n");
    for(i=0;i<r2;i++)
    {
        printf("\n");
        for(j=0;j<c2;j++)
        printf("%d\t",brr1[i][j]);
    }

```

```

    }
//multiplication of matrix
    for(i=0;i<r1;i++)
        for(j=0;j<c2;j++)
            crr1[i][j]=0;
            for(i=0;i<r1;i++)    //row of first matrix
                {
                    for(j=0;j<c2;j++)    //column of
second matrix
                        {
                            sum=0;
                            for(k=0;k<c1;k++)
                                sum=sum+arr1[i][k]*brr1[k][j];
                                crr1[i][j]=sum;
                        }
                }
    printf("\nThe multiplication of two matrices is :
\n");
    for(i=0;i<r1;i++)
        {
            printf("\n");
            for(j=0;j<c2;j++)
                {
                    printf("%d\t",crr1[i][j]);
                }
        }
    }
printf("\n\n");

```


10) Transpose of a matrix

- Code:

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

```
void main()
```

```
{  
int arr1[50][50],brr1[50][50],i,j,r,c;
```

```
printf("\n\nTranspose of a Matrix :\n");  
printf("-----\n");
```

```
printf("\nInput the rows and columns of the  
matrix : ");  
scanf("%d %d",&r,&c);
```

```
printf("Input elements in the first matrix  
:\n");  
for(i=0;i<r;i++)  
{  
for(j=0;j<c;j++)  
{  
printf("element - [%d],[%d] : ",i,j);  
scanf("%d",&arr1[i][j]);  
}  
}
```

```

printf("\nThe matrix is :\n");
    for(i=0;i<r;i++)
    {
        printf("\n");
        for(j=0;j<c;j++)
            printf("%d\t",arr1[i][j]);
    }

```

```

for(i=0;i<r;i++)
{
    for(j=0;j<c;j++)
    {
        brr1[j][i]=arr1[i][j];
    }
}

```

```

printf("\n\nThe transpose of a matrix is : ");
for(i=0;i<c;i++){
    printf("\n");
    for(j=0;j<r;j++){
        printf("%d\t",brr1[i][j]);
    }
}
printf("\n\n");
}

```

11) Determinant of a matrix

- Code:

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

```
void main()
```

```
{
```

```
int arr1[10][10],i,j,n;
```

```
int det=0;
```

```
printf("\n\nCalculate the determinant of a 3  
x 3 matrix :\n");
```

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

```
printf("Input elements in the first matrix  
:\n");
```

```
for(i=0;i<3;i++)
```

```
{
```

```
for(j=0;j<3;j++)
```

```
{
```

```
printf("element - [%d],[%d] : ",i,j);
```

```
scanf("%d",&arr1[i][j]);
```

```
}
```

```
}
```

```
printf("The matrix is :\n");
```

```
for(i=0;i<3;i++)
```

```
{
```

```

        for(j=0;j<3 ;j++)
            printf("% 4d",arr1[i][j]);
        printf("\n");
    }

```

```

    for(i=0;i<3;i++)
        det = det +
        (arr1[0][i]*(arr1[1][(i+1)%3]*arr1[2][(i+2)%3] -
arr1[1][(i+2)%3]*arr1[2][(i+1)%3]));

```

```

        printf("\nThe Determinant of the matrix is:
%d\n\n",det);
    }

```

12) Check for equal matrices

- Code:

```

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

```

```

void main()
{
    int arr1[50][50], brr1[50][50];
    int i, j, r1, c1, r2, c2, flag =1;

```

```

        printf("\n\nAccept two matrices and check
whether they are equal :\n ");

```

```

        printf("-----
-----\n");

```

```

printf("Input Rows and Columns of the 1st
matrix :");
scanf("%d %d", &r1, &c1);

printf("Input Rows and Columns of the 2nd
matrix :");
scanf("%d %d", &r2,&c2);
printf("Input elements in the first matrix
:\n");
for(i=0;i<r1;i++)
{
for(j=0;j<c1;j++)
{
printf("element - [%d],[%d] : ",i,j);
scanf("%d",&arr1[i][j]);
}
}
printf("Input elements in the second matrix
:\n");
for(i=0;i<r2;i++)
{
for(j=0;j<c2;j++)
{
printf("element - [%d],[%d] : ",i,j);
scanf("%d",&brr1[i][j]);
}
}

```

```

printf("The first matrix is :\n");
for(i=0;i<r1;i++)
{
    for(j=0;j<c1 ;j++)
        printf("% 4d",arr1[i][j]);
    printf("\n");
}
printf("The second matrix is :\n");
for(i=0;i<r2;i++)
{
    for(j=0;j<c2 ;j++)
        printf("% 4d",brr1[i][j]);
    printf("\n");
}
/* Comparing two matrices for equality */

if(r1 == r2 && c1 == c2)
{
    printf("The Matrices can be compared : \n");
    for(i=0; i<r1; i++)
    {
        for(j=0; j<c2; j++)
        {
            if(arr1[i][j] != brr1[i][j])
            {
                flag = 0;
                break;
            }
        }
    }
}

```

```

        }
    }
}
else
{ printf("The Matrices Cannot be compared
:\n");
  exit(1);
}
if(flag == 1 )
    printf("Two matrices are equal.\n\n");
else
    printf("But,two matrices are not equal\n\n");
}

```

13) Merging sorted arrays

- Code:

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

```

void merge2arrs(int *bgArr, int bgArrCtr, int
*smlArr, int smlArrCtr)
{
    if(bgArr == NULL || smlArr == NULL)
        return;
    int bgArrIndex = bgArrCtr-1,
        smlArrIndex = smlArrCtr-1,
        mergedArrayIndex = bgArrCtr + smlArrCtr -
1;

```

```

    while(bgArrIndex >= 0 && smlArrIndex >=
0) {
    if(bgArr[bgArrIndex] >=
smlArr[smlArrIndex]){
        bgArr[mergedArrayIndex] =
bgArr[bgArrIndex];
        mergedArrayIndex--;
        bgArrIndex--;
    } else {
        bgArr[mergedArrayIndex] =
smlArr[smlArrIndex];
        mergedArrayIndex--;
        smlArrIndex--;
    }
}
if(bgArrIndex < 0)
{
    while(smlArrIndex >= 0)
    {
        bgArr[mergedArrayIndex] =
smlArr[smlArrIndex];
        mergedArrayIndex--;
        smlArrIndex--;
    }
} else if (smlArrIndex < 0)
{
    while(bgArrIndex >= 0)
    {

```



```

        bgArr[mergedArrayIndex] =
bgArr[bgArrIndex];
        mergedArrayIndex--;
        bgArrIndex--;
    }
}
}

```

```

int main()
{
    int bigArr[13] = {10, 12, 14, 16, 18, 20, 22};
    int smlArr[6] = {11, 13, 15, 17, 19, 21};
    int i;
    //----- print large array -----
    printf("The given Large Array is : ");
    for(i = 0; i < 7; i++)
    {
        printf("%d ", bigArr[i]);
    }
    printf("\n");
    //----- print small array -----
    printf("The given Small Array is : ");
    for(i = 0; i < 6; i++)
    {
        printf("%d ", smlArr[i]);
    }
    printf("\n");
}

```

```
//----- print merged array -----
--
merge2arrs(bigArr, 7, smlArr, 6);
printf("After merged the new Array is :\n");
for(i = 0; i<13; i++)
{
    printf("%d ", bigArr[i]);
}
return 0;
}
```

Pointers

1) Basic Demonstration:

- Code: #include <stdio.h>

```
void main()
{
    int m=300;
    float fx = 300.60;
    char cht = 'z';

    printf("\n\n Pointer : Demonstrate the use
of & and * operator :\n");
    printf("-----\n");
    int *pt1;
    float *pt2;
    char *pt3;
    pt1= &m;
```

```

pt2=&fx;
pt3=&cht;
printf ( " m = %d\n",m);
printf ( " fx = %f\n",fx);
printf ( " cht = %c\n",cht);
printf("\n Using & operator :\n");
printf("-----\n");
printf ( " address of m = %p\n",&m);
printf ( " address of fx = %p\n",&fx);
printf ( " address of cht = %p\n",&cht);
printf("\n Using & and * operator :\n");
printf("-----\n");
printf ( " value at address of m =
%d\n",*(&m));
printf ( " value at address of fx =
%f\n",*(&fx));
printf ( " value at address of cht =
%c\n",*(&cht));
printf("\n Using only pointer variable :\n");
printf("-----\n");
printf ( " address of m = %p\n",pt1);
printf ( " address of fx = %p\n",pt2);
printf ( " address of cht = %p\n",pt3);
printf("\n Using only pointer operator :\n");
printf("-----\n");
printf ( " value at address of m =
%d\n",*pt1);

```

```

    printf ( " value at address of fx=
%f\n",*pt2);
    printf ( " value at address of cht=
%c\n\n",*pt3);
}

```

2) Adding Number using reference

- Code : #include <stdio.h>

```

long addTwoNumbers(long *, long *);

```

```

int main()
{

```

```

    long fno, sno, sum;

```

```

    printf("\n\n Pointer : Add two numbers using call
by reference:\n");

```

```

    printf("-----\n");

```

```

    printf(" Input the first number : ");
    scanf("%ld", &fno);
    printf(" Input the second number : ");
    scanf("%ld", &sno);
    sum = addTwoNumbers(&fno, &sno);
    printf(" The sum of %ld and %ld is %ld\n\n", fno,
sno, sum);
    return 0;

```

```

}
long addTwoNumbers(long *n1, long *n2)
{
    long sum;
    sum = *n1 + *n2;
    return sum;
}

```

3) Finding maximum number by reference

- Code:

```

#include <stdio.h>
#include <stdlib.h>
void main()
{
    int fno,sno,*ptr1=&fno,*ptr2=&sno;

    printf("\n\n Pointer : Find the maximum
number between two numbers :\n");
    printf("-----\n");

    printf(" Input the first number : ");
    scanf("%d", ptr1);
    printf(" Input the second  number : ");
    scanf("%d", ptr2);
    if(*ptr1>*ptr2)
    {

```

```

    printf("\n\n %d is the maximum
number.\n\n",*ptr1);
}
else
{
    printf("\n\n %d is the maximum
number.\n\n",*ptr2);
}}

```

✓ 4)Printing element of an array using printer

- Code:

```

#include <stdio.h>
int main()
{
    int arr1[25], i,n;
    printf("\n\n Pointer : Store and retrieve
elements from an array :\n");
    printf("-----\n");
    printf(" Input the number of elements to store
in the array :");
    scanf("%d",&n);

    printf(" Input %d number of elements in the
array :\n",n);
    for(i=0;i<n;i++)
    {

```

```

        printf(" element - %d : ",i);
        scanf("%d",arr1+i);
    }
    printf(" The elements you entered are : \n");
    for(i=0;i<n;i++)
    {
        printf(" element - %d : %d \n",i,*(arr1+i));
    }
    return 0;
}

```

5) Print all possible permutaions of string using pointer

- Code:

```

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

```

```

void changePosition(char *ch1, char *ch2)
{
    char tmp;
    tmp = *ch1;
    *ch1 = *ch2;
    *ch2 = tmp;
}

```

```

void charPermu(char *cht, int stno, int endno)
{
    int i;
    if (stno == endno)
        printf("%s ", cht);
    else
    {
        for (i = stno; i <= endno; i++)
        {
            changePosition((cht+stno), (cht+i));
            charPermu(cht, stno+1, endno);
            changePosition((cht+stno), (cht+i));
        }
    }
}

```

```

int main()
{
    char str[] = "abcd";
    printf("\n\n Pointer : Generate permutations of
a given string :\n");
    printf("-----\n");
    int n = strlen(str);
    printf(" The permutations of the string are :
\n");
    charPermu(str, 0, n-1);
}

```



```
printf("\n\n");  
return 0;
```

```
}
```

✓ 6) Calculate the size of the string using pointer

- Code :

```
#include <stdio.h>  
int calculateLength(char*);  
  
void main()  
{  
    char str1[25];  
    int l;  
    printf("\n\n Pointer : Calculate the length of  
the string :\n");  
    printf("-----  
-----\n");  
  
    printf(" Input a string : ");  
    fgets(str1, sizeof str1, stdin);  
  
    l = calculateLength(str1);  
    printf(" The length of the given string %s is :  
%d ", str1, l-1);  
    printf("\n\n");  
}
```

```

int calculateLength(char* ch) // ch = base
address of array str1 ( &str1[0] )
{
    int ctr = 0;
    while (*ch != '\0')
    {
        ctr++;
        ch++;
    }
    return ctr;
}

```

7) Finding factorial of a number using pointer

- Code:

```

#include <stdio.h>
void findFact(int,int*);
int main()
{
    int fact;
    int num1;
    printf("\n\n Pointer : Find the factorial
of a given number :\n");
    printf("-----\n");
    printf(" Input a number : ");
    scanf("%d",&num1);

    findFact(num1,&fact);
}

```

```

        printf(" The Factorial of %d is : %d
\n\n",num1,fact);
        return 0;
    }

```

```

void findFact(int n,int *f)
{
    int i;

    *f=1;
    for(i=1;i<=n;i++)
        *f=*f*i;
}

```

Functions

✓ 1) Simple structure of function of adding two numbers

- Code:

```

#include <stdio.h>

```

```

    int sum (int, int); //function declaration
    int main (void)
    {
        int total;
        printf("\n\n Function : a simple
structure of function :\n");
        printf("-----
-----\n");
    }

```

```
total = sum (5, 6); //function call
printf ("The total is : %d\n", total);
return 0;
}
```

```
int sum (int a, int b) //function definition
{
    int s;
    s=a+b;
    return s; //function returning a value
}
```

✓ 2) Finding square of a number using function

- Code:

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

```
double square(double num)
{
    return (num * num);
}
```

```
int main()
{
    int num;
    double n;
    printf("\n\n Function : find square of any
number :\n");
}
```

```

        printf("-----\n");

```

```

        printf("Input any number for square : ");
        scanf("%d", &num);
        n = square(num);
        printf("The square of %d is : %.2f\n", num, n);
        return 0;
    }

```

3)Odd or even number check

- Code:

```

#include <stdio.h>

```

```

//if the least significant bit is 1 the number is odd
and 0 the number is even

```

```

int checkOddEven(int n1)

```

```

{
    return (n1 & 1); //The & operator does a
    bitwise and,
}

```

```

int main()

```

```

{
    int n1;
    printf("\n\n Function : check the number is
    even or odd:\n");

```

```

        printf("-----\n");

```

```
printf("Input any number : ");  
scanf("%d", &n1);
```

```
// If checkOddEven() function returns 1 then  
the number is odd
```

```
if(checkOddEven(n1))  
{  
    printf("The entered number is odd.\n\n");  
}  
else  
{  
    printf("The entered number is even.\n\n");  
}  
return 0;  
}
```

4) Decimal number to binary equivalent

- Code:

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

```
long toBin(int);
```

```
int main()
```

```
{
```

```
    long bno;
```

```
    int dno;
```

```
    printf("\n\n Function : convert decimal to  
binary :\n");
```

```

        printf("-----\n");
        printf(" Input any decimal number : ");
        scanf("%d",&dno);
        bno = toBin(dno);
        printf("\n The Binary value is : %ld\n\n",bno);

        return 0;
    }
    long toBin(int dno)
    {
        long bno=0,remainder,f=1;
        while(dno != 0)
        {
            remainder = dno % 2;
            bno = bno + remainder * f;
            f = f * 10;
            dno = dno / 2;
        }
        return bno;
    }
}

```

5) Checking for Armstrong or perfect number

- Code:

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

```

int checkArmstrong(int n1);
int checkPerfect(int n1);

```

```

int main()
{
    int n1;
    printf("\n\n Function : check Armstrong and
perfect numbers :\n");
    printf("-----\n");

    printf(" Input any number: ");
    scanf("%d", &n1);

    //Calls the isArmstrong() function
    if(checkArmstrong(n1))
    {
        printf(" The %d is an Armstrong
number.\n", n1);
    }
    else
    {
        printf(" The %d is not an Armstrong
number.\n", n1);
    }

    //Calls the checkPerfect() function
    if(checkPerfect(n1))
    {

```



```

        printf(" The %d is a Perfect number.\n\n",
n1);
    }
    else
    {
        printf(" The %d is not a Perfect
number.\n\n", n1);
    }
    return 0;
}

```

// Checks whether a three digits number is Armstrong number or not.

//An Armstrong number is an n-digit number that is equal

//to the sum of the n-th powers of its digits.

```
int checkArmstrong(int n1)
```

```
{
    int ld, sum, num;
```

```
    sum = 0;
```

```
    num = n1;
```

```
    while(num!=0)
```

```
    {
```

```
        ld = num % 10; // find the last digit of the
number
```

```
        sum += ld * ld * ld; //calculate the cube of
the last digit and adds to sum
```

```
        num = num/10;
```

```

    }
    return (n1 == sum);
}
// Checks whether the number is perfect number
or not.
//a perfect number is a positive integer that is
equal to
//the sum of its positive divisors excluding the
number itself
int checkPerfect(int n1)
{
    int i, sum, num;
    sum = 0;
    num = n1;
    for(i=1; i<num; i++)
    {
        /* If i is a divisor of n1 */
        if(num%i == 0)
        {
            sum += i;
        }
    }
    return (n1 == sum);
}

```

1) Simple Declaration and implementation

Code:

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

```
struct Books {
    char title[50];
    char author[50];
    char subject[100];
    int book_id;
};
```

```
int main( ) {
```

```
    struct Books Book1;    /* Declare Book1 of type
Book */
```

```
    struct Books Book2;    /* Declare Book2 of type
Book */
```

```
    /* book 1 specification */
```

```
    strcpy( Book1.title, "C Programming");
```

```
    strcpy( Book1.author, "Nuha Ali");
```

```
    strcpy( Book1.subject, "C Programming Tutorial");
```

```
    Book1.book_id = 6495407;
```

```
    /* book 2 specification */
```

```
    strcpy( Book2.title, "Telecom Billing");
```

```
    strcpy( Book2.author, "Zara Ali");
```

```
    strcpy( Book2.subject, "Telecom Billing Tutorial");
```

```
Book2.book_id = 6495700;
```

```
/* print Book1 info */  
printf( "Book 1 title : %s\n", Book1.title);  
printf( "Book 1 author : %s\n", Book1.author);  
printf( "Book 1 subject : %s\n", Book1.subject);  
printf( "Book 1 book_id : %d\n", Book1.book_id);
```

```
/* print Book2 info */  
printf( "Book 2 title : %s\n", Book2.title);  
printf( "Book 2 author : %s\n", Book2.author);  
printf( "Book 2 subject : %s\n", Book2.subject);  
printf( "Book 2 book_id : %d\n", Book2.book_id);
```

```
return 0;
```

```
}
```

2) Storing many employee data using structure

- Code:

```
#include<stdio.h>  
#include <string.h>  
struct employee  
{   int id;  
    char name[50];  
    float salary;  
}e1,e2; //declaring e1 and e2 variables for  
structure  
int main( )
```

```

{
    //store first employee information
    e1.id=101;
    strcpy(e1.name, "Sonoo Jaiswal");//copying
string into char array
    e1.salary=56000;

    //store second employee information
    e2.id=102;
    strcpy(e2.name, "James Bond");
    e2.salary=126000;

    //printing first employee information
    printf( "employee 1 id : %d\n", e1.id);
    printf( "employee 1 name : %s\n", e1.name);
    printf( "employee 1 salary : %f\n", e1.salary);

    //printing second employee information
    printf( "employee 2 id : %d\n", e2.id);
    printf( "employee 2 name : %s\n", e2.name);
    printf( "employee 2 salary : %f\n", e2.salary);
    return 0;
}

```