Date:22-7-2025

Self Practice Exercises

Beginners C-Programs list

the soft copies of this programs-list & c-material is available in

https://srihari1973.github.io/cfamily.site/c%20programs%20list(2024).pdf

Since 1999

C-Family Computer Education

Flyover Pillar No:16, Service Road, Benz Circle

Vijayawada, AP, India, pincode-5200109440-030405, 8500-117118

about printf() statement

syntax: printf(" some text message with format strings " , list of values);

the format strings are: %d , %f , %ld , %c , %s , %lf , etc; use %d for int , %f for float , %ld for long int
the format strings such as %d , %f , %c, etc tells the conversion of values from binary to decimal and decimal to binary while scanning and printing values. These are internally used by the scanf() and printf().

```
int A = 10 , B = 20;
```

1. printf(" %d %d " , A , B); → ?
2. printf(" A B "); → ?
3. printf(" %d %d " , A+1 , 2+B); → ?
4. printf(" %d+1 %d+2 " , A , B); → ?
5. printf(" %d + %d " , A , B); → ?
6. printf(" %d " , A+B); ?
7. printf(" A+B = %d " , A+B); ?
8. printf(" A+B is " , A + B); ?
9. printf(" A is %d , B is %d " , A , B); → ?
10. printf(" A(%d) , B(%d) " , A , B); → ?
12. printf(" A = %d , B = %d " , A , B); → ?
13. printf(" output = %d " , A+B); ?
14. printf("\n %d + %d = %d" , A , B , A+B); ?
15. printf("\n %d - %d = %d" , A , B , A-B); ?
16. printf("\n %d < %d = %d" , A , B , A<B); ? // A<B → 10<20 → true → 1
17. printf("\n %d > %d = %d" , A , B , A>B);? // A>B → 10>20 → false →0
18. printf("\n %d == %d = %d" , A , B , A==B); ? // A==B → 10==20 → false → 0

we can add **'width'** to the format strings, for example: %5d , %7d , %05d , %f , %.2f , %05.2f, etc.

%7d → here 7 is said to be the maximum width of output value which we want to show on the screen.

eg: printf("%7d", 523); → BBBB523 // Here B means blank space, added by printf()

if width > output-value size then spaces are added before the value by the printf() statement.

if width < digits in value then no spaced are added, it prints normally (here ignores the width).

Let us see some examples,

```
printf("%7d", 523);      → BBBB523      // Here B means blank space, added by printf()
printf("%07d", 523);     → 0000523      // pads with zeros instead of spaces
printf("%02d", 7234);    → 7234         // here width < digits in value, so prints normally ( no affect )
printf("%.2f", 8671.4256); → 8671.42    // .2 is cutoff decimal values
printf("%7.3f", 823.4); → BBBB823.400
printf("%7.2f", 823.423); → BBBB823.42
printf("%7.0f", 823.4); → BBBB823
```

about scanf() statement

Let A , B , C are int-type variables, and also assume 34 , 28 23 are the input of A, B, C in the following example, then the user should be typed in the keyboard as given below

1) scanf("%d%d" , &A , &B); ← 34 space 28 ↲ // space is the default separator between two values

2) scanf("%d%d" , &A , &B); ← 34 ↲ 28 ↲ // enter-key can also be used to separate two values

3) scanf("%d%d%d" , &A , &B, &C); ← 34 28 23 ↲ // scanning three values

4) scanf("%d" , &A); ← 34 ↲ // scanning single integer value

5) scanf("%f" , &X); ← 34.45 ↲ // scanning single float value (let X , Y are float variables)

6) scanf("%f%f" , &X , &Y); ← 34.45 85.67 ↲ // scanning two float values

syntax of scanf() statement

Syntax: scanf("format strings" , list of variable addresses)

7. scanf("%d%d", &A , &B); ← 34 space 28 ↲ // space is the default separator between two values

8. scanf("%dQQQ%d", &A , &B); ← 34QQQ28↲ // here QQQ is the separator for 34 and 28 values

9. scanf("%d/%d", &A , &B); ← 34/28↲ // here slash(/) is the separator for 34 and 28 values

10. scanf("%d,%d", &A , &B); ← 34,28↲ // here comma is the separator for 34 and 28 values

conclusion: other than %d should be given as it is, let us see following example

11. scanf("Hello%dWorld%d", &A , &B); ← Hello34World28 ↲

12. scanf("XY%dZ", &A); ← XY34Z↲

13. scanf(" %d ", &A); ← space34space↲

note: Suggestion is, do not give any extra symbols including spaces in the scanf().
It should be like "%d%d" should not be like " %d %d " [this suggestion is for scanf() and not for printf()]

1) Demo program, finding average of two integer values

```
#include<stdio.h>
int  main()
{       int  A , B ;
        float average;
        A=10;  B=20;            // two instructions can be given in same line separated by coma or semi-colon
        average=(A+B)/2;     //  should not be like : average = A+B/2;
        printf(" average value is  %f",  average );
}
```

2) Finding the difference of two scanned integer values

```
#include<stdio.h>
int  main()
{       int  x , y , diff ;
        printf("enter two  values :");
        scanf("%d%d" ,  &x, &y);
        diff=x-y;
        printf("output is: %d",  diff );
}
```

keyboard
enter two value: 23 13 ↵

output is: 10

3) Finding the difference of two scanned float values

```
#include<stdio.h>
int  main()
{       float  x , y , diff ;
        printf("enter  two  float values :");
        scanf("%f%f" ,  &x, &y);
        diff=x-y;
        printf("output is: %f",  diff );
}
```

keyboard
enter two float value: 23.4 13.2 ↵

output is: 10.2

4) Demo program, how to use pow() and sqrt() math functions.

pow(x,y) → this finds x^Y, here X is base, Y is exponent.

sqrt(x) → this finds square root of x (√x)

```
#include<math.h>     // when  pow() or sqrt() function is used , this "math.h" file should be included
#include<stdio.h>    // when printf() or scanf() function is used,  this "stdio.h" file should be included
int  main()
{       int  A , B;
        A = pow(2,10);      // A=2^10
        B = sqrt(16);
        printf(" A = %d  ,  B = %d ", A , B );     //  output:  A=1024 , B=4
}
```

5) Finding sum of equation $5x^3 + 2x + 10$, here 'x' is input value

ip: if x is 3

op: 5*x*x*x +2*x+10 → 5*3*3*3+2*3+10 → 135 + 6 + 10 → 151

```
#include<stdio.h>
int main()
{       int  x , y;
        printf("enter x value :");
        scanf("%d", &x);
        y=5*x*x*x+2*x+10;          // we can't write X^3 in the program, we need to write as x*x*x
        printf("output is: %d",  y );
}
```

keyboard
enter x value: 3↵

Note: for bigger values like x^{10} use pow() function, but for smaller values like x^3 write as x*x*x.
because, the pow() function takes much machine code.

6) Demo, printing multiplication table up to 10 terms

ip: **if N is 7**

op: 7*1=7
7*2=14

7*10=70

```
int main()
{       int N;
        printf("enter table Number::");
        scanf("%d", &N);
        printf("\n %d * %d = %d", N, 1, N*1);      → 7 * 1 = 7
        printf("\n %d * %d = %d", N, 2, N*2);      → 7 * 2 = 14
        printf("\n %d * %d = %d", N, 3, N*3);      → 7 * 3 = 21
        -----
        printf("\n %d * %d = %d", N, 10, N*10);    → 7 * 10 = 70
}
```

keyboard
enter table Number : 7 ↵

7) Demo program adding total number of apples ate by 3 persons. (using only 2 variables)

```
Int  main()
{       int N, sum;
        printf("enter how many apples Rama had");
        scanf("%d", &N);
        sum=N;
        printf("enter how many apples Sita had:");
        scanf("%d", &N);
        sum=sum+N;
        printf("enter how many apples Laxmi had:");
        scanf("%d", &N);
        sum=sum+N;
        printf("total apples count is %d",  sum );
}
```

input: 3

input:5

input: 7

8) Demo program how the assignment operator(=) works in programs

```
int main()
{   int A,B,C;      // here memory space creates for A , B, C.  Each takes 2 byte. (total 6 bytes )
    A=10;           // here the operator(=) puts 10 into A
    B=A;            // here the operator(=) copies A value to B, so after copying, B gains same value of A
    printf("A is %d, B is %d" , A , B );

    A=10;
    A=A+1;          // here A+1 is assigning to A itself, so A increments to 11.
    A+1;            // this is incomplete statement, here A+1 is not assigning to any variable, so compiler ignores this code
                    // this code will not be executed
    A=10;
    A/2;            // this is also incomplete statement, compiler ignores, and this code will not be executed

    A=10;
    B=A;
    A++;
    printf("A is %d , B is %d ", A , B );

    A=B=C=200;      // copies 200 into all A, B, C
    printf("A , B , C is %d %d %d", A , B , C );
}
```

9) Demo program for pre/post increment operators (++/--)

```
int A=10, B=10;
printf("%d  %d ", A , B );
B = ++A;
```

First increment A value by 1 and then assigns to B, this is as

```
++A;
B=A;
```

```
A=10;
printf("%d  %d ", A , B );
B = A++;
printf("%d  %d ", A , B );
```

First assign A value to B and then increment A by 1, this is as

```
B=A:
A++;
```

```
A = 10;
B = ++A*2;
printf("%d  %d", A, B);
```

```
++A;
B=A*2;
```

```
B = 2*A++;
printf("%d  %d", A, B);
```

```
B=2*A;
A++;
```

```
A = 1;
B = ++A*2  +  ++A*3  +  ++A*4;        // pre means: increment and then substitute the value
printf("%d  %d", A, B);               // post means: substitute the value and then increment

A = 1;
B = ++A*2  +  ++A*3  +  A++*4  +  A++*5  +  ++A;
printf("%d  %d ", A , B );
```

10) Demo program on manipulating digits in a number

The operator '/' gives quotient of division, whereas the operator '%' gives remainder. For example

2345/10 → 234 2345%10 → 5 (234/10)%10 → (23)%10 → 3
2345/100 → 23 2345%100 → 45 (234%100)/10 → (34)/10 → 3
2345/1000 → 2 2345%1000 → 345 73%10 → 3
2345/10000 → 0 2345%10000 →2345 7%10 → 7

```
int main()
{       int  k , n = 2345;
        k = n/100+n%100;
        printf("\n sum = %d", k);        ?

        k = n%10+n/1000;
        printf("\n sum = %d", k);        ?

        k = (n/10)%10 + (n/100)%10;
        printf("\n sum = %d",  k );      ?
 }
```

11) sum of digits in a given 3 digit number

ip: 345
op: 3+4+5 → 12

```
#include<stdio.h>
int main()
{       int  N ,sum ;
        printf("enter 3 digit value :");
        scanf("%d" , &N );
        sum=N%10;            // gives last digit  5 as remainder from 345

        N=N/10;              //  removes last digit 5 from N, here after N becomes 34
        sum=sum+N%10;        // adding second digit  4 to sum

        N=N/10;              // removing second digit 4 from N, here after N becomes 3
        sum=sum+N%10;        // adding first digit  3  to N
        printf("output is: %d",  sum );
}
```

12) Above program in simplified form.

ip: 345
op: 3+4+5 → 12

```
#include<stdio.h>
int main()
{       int  N ,sum ;
        printf("enter 3 digit value :");
        scanf("%d" , &N );
        sum = N%10  +  (N/10)%10  +  N/100;
        printf("output is: %d",  sum );
}
```

15)
```
int A=100;
A=A*2;
printf("%d ", A );
A=A*2;
printf("%d ", A );
```

16)
```
int A=50;
A=A/2;
printf("%d ", A );
A=A/2;                  hint: int/int → int
printf("%d ", A );
```

17)
```
float  A;
A=5/2;                  ( int/int → int )
printf(" %f ",  A );

A=5.0/2;                ( float/int → float )
printf(" %f ",  A );
```

18)
```
float  A;
A=1.0 * 5/2;            ( float * int → float )
printf(" %f  ",  A );

A=5/2 * 1.0;
printf(" %f  ",  A );
```

19)
```
a=10;     b=20;
a=b;
b=a;
printf("%d    %d",  a , b);
```

20)
```
a=10, b=20;
t=a;
a=b;
b=t;
printf("%d    %d",  a , b);
```

21)
```
b = 100;
b = a = b/2;
printf("%d  %d", a , b );
a = a + b;
b = a + b;
printf("%d  %d", a , b );
```

22)
```
a = 1;
printf("%d ", 7*a );
a++;
printf("%d ", 7*a );
a++;
printf("%d ", 7*a );
```

23)
```
a=12;
b = ++a;
printf("%d   %d",  a,  b);
b = a++;
printf("%d   %d",  a,  b);
```

24)
```
a = 12;
b = 1+a;
printf("%d   %d",  a,  b);
b = ++a;
printf("%d   %d",  a,  b);
```

25)
```
int a,b;
a=b=100;
a++;
printf("%d   %d",  a,  b);
```

26)
```
A = 12;
printf("%d  %d",    A ,   -A );
A = -A;
printf("%d  %d",    A ,   -A );
```

27)
```
a=7%6;
printf("%d", a );
a=7%7;
printf("%d", a );
a=7%8;
printf("%d", a );
```

28)
```
printf("%d",   2+4*5+1 );
printf("%d",   2*4+4/2 );
printf("%f",   4.0/2*2 );
printf("%f",   4.0/(2*2) );
printf("%f",   20.0/4/2 );
```

29)
```
a=234;
printf("%d ",  a%10 );
printf("%d ",  a%100 );
printf("%d ",  a%1000 );
```

30)
```
a=2345;
printf("%d",  a/10 );
printf("%d",  a/100 );
printf("%d",  a/1000 );
```

31)
```
a=2345;
a=a%10 + a/1000;
printf("%d ", a );
```

32)
```
a=2345;
a = (a%100) + a/100;
printf("%d ", a );
```

33)
```
a=345;
printf("%d",   a%10 );
printf("%d",  ( a/10)%10 );
printf("%d",   a/100 );
```

34)
```
a=345;
printf("%d",   a/100 );
printf("%d",  (a%100)/10 );
printf("%d",   a%10 );
```

35) Write a program to accept 3 values from keyboard and print total and average of them

ip: 10 20 30

op: total is 60 , average is 20.00

step1: take variable names as a, b, c, total, avg.

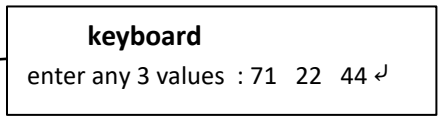
step2: scan 3 values into **a, b, c**.

step3: find sum of a+b+c and store into **'total'**

step4: find average of (a+b+c)/3 and store into **'avg'**

step5: print **'total'** and **'avg'** on the screen.

```
#include<stdio.h>
int main()
{   int a,b,c, total;
    float avg;
    printf("enter any 3 values :");
    scanf("%d%d%d" , &a, &b, &c);
    ----
    ----
    printf("total is %d \n average is %f", total , avg);
}
```



---

36) Write a program to accept a value (N) from keyboard and print its opposite sign value.

ip: 19 ip: -19 ip: -40

op: -19 op: 19 op: 40

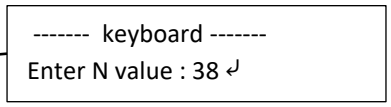
step1: take **N** as **int** type variable

step2: scan **N** from keyboard

step3: multiply **N** with **-1** to get opposite sign value and store result back to **'N'** itself

step4: print **N** as output.

```
#include<stdio.h>
int main()
{   int  N;
    printf("enter N value:");
    scanf("%d" , &N);
    --------
    printf(" opposite sign value is ",  N );
}
```



---

37) if two digit number like 47 is entered through the keyboard then print sum of digits (4+7→11)

ip: 47 ip: 84

op: 4+7 → 11 op: 8+4 → 12

logic: divide the input N with 10, and collect the remainder & quotient.

```
#include<stdio.h>
int main()
{   int  N, sum;
    printf("enter two digit single number:");
    scanf("%d" , &N);
    --------
}
```

10) 47 (4 →quotient 40 ---------- 7 → remainder	sum = N/10 + N%10; sum = 47/10 + 47%10; sum = 4 + 7

---

38) If single pen cost is rs:3/-, then find cost of N pens.
Now write a program to accept **N** as input and print total cost of **N** pens.

input: 10 input: 40
output: 30 output: 120

step1: take variable name '**N'** to hold input value. (here **'N'** is no.of pens)
step2: take variable name as '**totalCost'** to store total cost of **'N'** pens (output value)
step3: scan number of pens as '**N'** // cost of each pen is fixed value rs:3/-, so no need to scan the cost
step4: calculate N*3 as total cost of N pens and store result into **'totalCost'**
step5: print '**totalCost'** of N pens

```
#include<stdio.h>
int main()
{   int  N , totalCost ;
    printf("enter number of pens available :");
    scanf("%d" , &N);
    -------
    printf("cost of %d pens is %d rupees",  N,  totalCost );
}
```

---

39) Find Fahrenheit from given Celsius. (formula is: F = 9.0/5*C+32)
scans Celsius from keyboard and prints Fahrenheit as output. Take C, F as float variables.

ip: enter Celsius: 38
op: Fahrenheit is: 100.4

---

40) The BigBite shop owner offers half the number of eggs as free for every purchase of 'N' eggs.
For example, if a customer purchases 2 eggs, he gets 1 egg free; if he purchases 10 eggs, he gets 5 eggs free.
Now scan the number of eggs (N) purchased by the customer and print how many eggs should be delivered.

input: 10 input: 2 input: 25
output: 15 (here 5 is free) output: 3 output: 25+12→37

take variables: 'N' as no.of eggs purchased by customer, 'T' as no.of eggs to be delivered with free eggs.
here 'N' as input variable, ' T' as output variable.

---

41) A fruit vendor is selling apples at a cost of ₹7 each. If a customer wants to buy apples for M rupees, how many apples can he get, and how much change will he get back?

ip: **M**=20rs then he get 2 apples and he get back 6/- rupees change.

step1: take variable names as M, applesCount, moneyChange;
step2: scan M as money spending by the customer
step3: calculate how many apples he get (M/7 gives quotient as no.of apples)
step4: calculate money change he get back. (M%7 gives remainder as money change)
step5: finally print apples count and change of money.

---

42) Code to accept radius of circle and find area and perimeter

Circle area is: πr^2

Circumference is: $2 \pi r$

Note: The symbol **pie** (π) is not available in the keyboard , beside C-Language does not know such symbols, so use 3.14 in place of π in the program. For example: area = 3.14*radius*radius;

ip: enter radius: 5 ip: enter radius: 8

op: circle area is 78.5 , circumference is 31.4 op: circle area is 200.96 , circumference is 50.24

take variable names: radius, area, perimeter; (take all float types)

---

43) Code to accept area of circle and find radius (input is area, output is radius)

ip: area: 78.5

op: radius of circle is: 5

calculations: radius = sqrt(area/3.14)

note: here sqrt() is a predefined library function to calculate square root value

when we use this function, we need to add header file "#include<math.h>"

---

44) Code to accept two values into variables (X,Y) and print after swapping them (exchanging values).

ip: x=12 y=34

op: x=34 y=12

Note: For swapping, one might write the code as Y = X; and X = Y;. However, when Y = X is executed first, the value of Y is replaced with the value of X, and the original Y value is lost. So, this code gives the wrong output. The following procedure provides the correct approach for swapping two values.

step1: take extra variable (temp) to exchange values.

step2: store X-value to temp // that is, saving X-value in temp.

step3: store Y-value to X // replacing X with Y-value.

step4: store temp-value to Y // replacing Y with X-value, here temp contains original X-value.

---

45) A shopkeeper is selling items with a 12% discount. Now, your task is to scan the item price from the keyboard and print the final price after the discount.

ip: enter item price: 1500

op: item price = 1500.00, discount=180.00, final price=1320.00

step1: take variable names as: item<u>P</u>rice, discount, final<u>P</u>rice

step2: scan item<u>P</u>rice from K.B // K.B→ Keyboard

step3: calculate discount // discount = itemPrice*12.0/100

step4: calculate final<u>P</u>rice // finalPrice = itemPrice – discount

step5: print all (item<u>P</u>rice , discount , final<u>P</u>rice)

---

46) Code to accept employee basic salary from keyboard and print net salary as shown below.

HRA→24% on basic salary (HRA→House Rent Allowance)

TA→ 10% on basic salary (TA→Travelling Allowance)

net salary = basic salary + HRA + TA;

step1: take variable names as: basic**S**alary, net**S**alary, HRA, TA (take **float**-types)

step2: read basic**S**alary as input

step3: calculate HRA

step4: calcucate TA

step5: find and print net**S**alary

---

47) Ramu ate some apples in the morning, afternoon, and evening. Now scan how many apples he ate in the morning, afternoon, and evening, and finally print the total count. Try this program using only 2 variables.
Hint: Do not change the existing code; just guess the dotted lines.

```
int main()
{   int n, total;
    printf("enter no.of apples he had today morning :");
    scanf("%d", &n);
    ----------------------      // here copy/assign 'n' to 'total'
    printf("enter no.of apples he had today afternoon :");
    scanf("%d", &n);
    ----------------------      // here add 'n' to 'total' , 'total' already contained morning apples count
    printf("enter no.of apples he had today evening :");
    scanf("%d", &n);
    ----------------------      // here add 'n' to 'total'
    printf("total apples he had is : %d", total );
}
```

---

48) 'M' is the amount to be shared among three persons: A, B, and C. A takes 70% of M, B takes 60% of the remaining amount after A's share, and the leftover amount is taken by C.
Write program to print share of each person.
if M is 100/-, then A gets 70/- , B gets 18/-, C gets 12/-. [70% of M is calculated as: A = 70.0*M/100]
if M is 200/-, then A gets 140/-, B gets 36/-, C gets 24/-

---

49) Ramu has 'X' kg of tomatoes and 'Y' kg of beans. The cost of tomatoes is ₹5 per kg, and the cost of beans is ₹12 per kg. Ramu wants to sell all these vegetables and buy some apples. If each apple costs ₹8, then how many apples can he roughly get from the money he earns by selling the vegetables?
The input is: X, Y and the output is: applesCount.
input : Enter how many kg of tomatoes and apples he got : 10 15 ↵
output: he get 28 apples approximately

---

50) Ramu has 'D' Dollars and 'R' Rupees. This amount is to be shared equally between two people, say P1 and P2. How much money does each person get? (1 Dollar = 82 Rupees). Person P1 takes the money in Dollars, and P2 takes it in Rupees. Let D and R be the input values, and p1 and p2 be the output values.
ip: 20 1200 (20 Dollars , 1200 Rupees)
op: 17/- Dollors for p1 , 1420/- Rupees for p2 (approximate values)

---

51) The cost of an apple is ₹7, an orange is ₹3, and a banana is ₹1. Mr. Ramu wants to spend M rupees to buy these fruits. How many maximum apples, then oranges, and finally bananas can he get?"
ip: M=25 op:Apples=3, Oranges=1, Bananas=1
ip: M=27 op:Apples=3, Oranges=2, Bananas=0
ip: M=28 op:Apples=4, Oranges=0, Bananas=0

```
step1: take M , applesCount, orangesCount, bananasCount;
step2:  scan Money(M) from K.B
step3: no.of apples he get is M/7
step4: M=M%7;       // reducing M to remaining money which is in hand after purchasing apples
step5: here calculate oranges count, how many he get
---------------
```

---

52) Converting H:M:S format time into seconds format.

Code to accept a time in H:M:S format as input and print output in seconds format.

Hint: Here we have to scan time like 3 values, for example: scanf("%d%d%d", &H, &M, &S);

ip: 2 40 50 (2-hr , 40-min , 50-sec) ip: 0 2 50 (0-hr , 2-min, 50-sec)

op: 9650 seconds op: 170 seconds

Process: total seconds is: N = H*3600 + M*60 + S // 1 hour=3600 seconds, 1 minute=60 seconds

---

53) converting seconds format time into H:M:S format

Mr. Ramu took N seconds to reach his school. Convert this time into hours, minutes, and seconds.

For example, if N is 7490, the output should be 2:4:50. The picture below shows how to calculate H, M, S.

ip: 7490

op: 2 : 4 : 50

hint: one minute has 60 seconds, and one hour has 3600 seconds.

step1: scan the number of seconds taken by Ramu (input as N).

step2: h=N/3600; // here we get no.of hours in N

step3: N=N%3600; // cutting N to remaining seconds which is left after taking hours from N

m=N/60; // taking minutes in N

---

printf("%d : %d : %d", h, m, s);

---

54) Code to accept two shifts of working time of an employee and find total time worked in two shifts.

ip: 12 50 58 (take variables as H1, M1, S1)

2 53 55 (take variables as H2, M2, S2)

op: 15 : 44 : 53 (total duration worked in two shifts)

```
void main()
{  int h1,m1,s1,   h2,m2,s2,   h3,m3,s3;
   printf("Enter shift1 duration time:");
   scanf("%d%d%d", &h1, &m1, &s1);
   printf("Enter shift2 duration time:");
   scanf("%d%d%d", &h2, &m2, &s2);
   // many people write following logic, but it is wrong,
   s3=s1+s2;      // this addition may cross 60 seconds , so it is wrong time
   m3=m1+m2;      // this addition may also cross 60 minutes
   h3=h1+h2;
```

procedure for adding two values.

1) convert two times into seconds. (like above said problems)
2) add these two times which are in seconds now.
3) distribute total seconds into H:M:S format. // distribute this seconds N value to h3,m3,s3
4) print output time. // print(h3, m3, s3)

---

55) Code to accept two complex numbers from keyboard and print sum of them (also try product)

let us take variable names: real1,img1 , real2,img2, real3,img3

ip: 4 5 ↵ (4+5i) // scanf("%d %d", &real1, &img1);

6 7 ↵ (6+7i) // scanf("%d %d", &real2, &img2);

op: 10+12i // real3=real1+real2 and also img3=img1+img2;

// print output as→ printf(" %d + %d i ", real3, img3);

---

56) Code to accept four digit number(N) like 3456 and print sum of first & last two-digits (34+56→90)

57) Code to accept four digit number(N) like 3456 and print sum of first & last digit (3+6→9)

58) If **'N'** has 3 digits like 347, then print sum of 3 digits (3+4+7→14)

59) Code to accept four digit number(N) like 3456 and print middle digits (45)

60) Code to accept four digit number(N) like 3456 and print sum of middle digits (4+5 →9)

61) Code to accept three digit number(N) like 345 and print its reverse (generate as: 5*100+4*10+3*1=543)

Home Work

70) Write a program to accept 4-digit single number(N) from keyboard and print sum of all digits.

ip: 4567

op: 4+5+6+7 → 22

Process: Extract digit by digit from N and add to 'sum' variable. The logic is as follows

step1. Divide N with 10 and take the remainder, the remainder is always the last-digit of N when a number divide with 10, for example 4567%10 → 7, add this 7 to variable 'sum'.

step2. To get next digit 6 from 4567, now remove current last-digit 7 from N, this is by doing N=N/10. after N=N/10, the N value becomes 4567/10 → 456

step3. Repeat above step1 & step2 for 4 times for adding all digits.

---

71) Write a program to accept 4-digit binary value N and print equaling decimal value.

ip: 1101

op: $(1*2^3) + (1*2^2) + (0*2^1) + (1*2^0)$ → 8 + 4 + 0 + 1 → 13

1	1	0	1
2^3	2^2	2^1	2^0

→ $1*2^3 + 1*2^2 + 0*2^1 + 1*2^0$ → 8 + 4 + 0 + 1 → 13

step1. divide N with 10 and get the last digit as remainder, here remainder of 110<u>1</u> is → (1)
multiply this remainder '1' with 2^0 and add to 'sum'. This is like sum=sum+(n%10*2^0)

step3. to get next-digit of N, remove current last-digit from N, by doing N=N/10, [1101/10→110]

step4. repeat these steps for 4 times.

We can't take or type values 2^0, 2^1, 2^2, 2^3, 2^4 … directly in the computer, so take these values as 1, 2, 4, 8,…

---

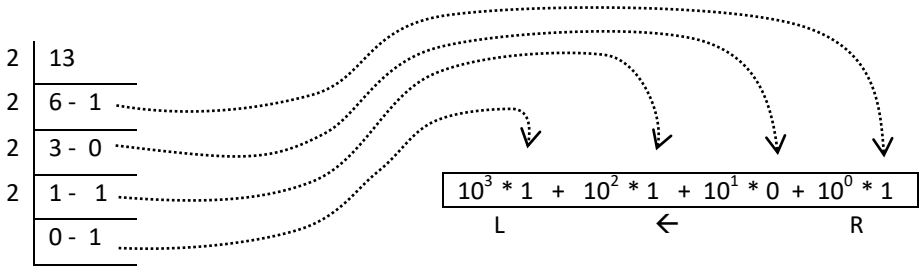
72) Write a program to find binary number from a given decimal number (Let the input is below 15)

step1: divide N with 2 and take the remainder

step2: now multiply this remainder with 10^0 and add to variable 'sum' [sum = sum + n%2*10^0]

step3: now cut down N to N/2 as shown in the picture. [13/2 →6]

step3: repeat above steps for 4 times. [here for values 10^0 , 10^1 , 10^2… take as 1 , 10 , 100 …]



---

73) If a single digit(N) entered through the keyboard then write a program to print next digit.

note: if input is 7 then output is 8, if input is 8 then output is 9, but if input is 9 then output is 0.

(use % operator to get result)

---

74) If a single digit(N) entered through keyboard then write a program to print next digit.

note: if input is 7 then output is 8, if input is 8 then output is 9, but if input is 9 then output is 1.

---

75) If a four-digit number is input through the keyboard, write a program to print a new number by adding one to each of its digits. If the number is 2391 then the output should be displayed as 3502

---

76) Code to accept only four digit number(N) like 3456 and print reverse of it (6543)

---

77) If a four-digit number(N) is input through the keyboard, write a program to swap first and last digit and print them. (Let us assume last digit of input N is not zero)

ip: 3456 ip: 3778

op: 6453 op: 8773

---

78) Code to accept two numbers as numerator(N) & denominator(D) and print remainder without using modulus operator (%). For example, if input N=22 & D=4 then remainder is 2.

Hint: use operators * , - and /

Note: the expression: 22/4 → 5 (not 5.5)

Logic: to get remainder, the equation is N-N/D*D → 22-22/4*4 → 22-20 → 2

---

79) code to find simple interest(si) from a given principle, rate of interest and time

ip: say principle is 1000, rate of interest is 2.00, time is 12 (12months)

op: simple interest is = 240.00

Process: step1: scan(p, t, r) as input values.

step2: calculate si=p*t*r/100.

step3: print(si)

---

if-else

Guess the output of following programs

1)

```
void main()
{
    if( 10 > 5 )
    {   printf(" A ");
        printf(" B ");
    }
    printf(" C ");

    if( 10 < 5 )
    {   printf(" D ");
        printf(" E ");
    }
    printf(" F ");

    if( 10 == 5 )
    {   printf(" G ");
        printf(" H ");
    }
}
```

op: ?

2)

```
void main()
{
    if( 10 > 5 )
    {   printf(" A ");
        printf(" B ");
    }
    else
    {   printf(" C ");
        printf(" D ");
    }
    printf(" E ");

    if( 10 < 5 )
    {   printf(" F ");
        printf(" G ");
    }
    else
    {   printf(" H ");
        printf(" i ");
    }
    printf(" j ");
}
```

op: ?

3)

```
if(a==10)
{   printf("red");
    if(b==20)
    {   printf("green");
    }
    else
    {   printf("blue");
        printf("white");
    }
    printf("black");
}
printf("yellow");
```

ip: a is 10, b is 20
op: ?

ip: a is 10, b is 30
op: ?

ip: a is 20, b is 20
op: ?

4)

```
if(a==10)
{   printf("red");
    if(b==20)
    {   printf("green");
    }
    else
    {   printf("blue");
        printf("white");
    }
    printf("black");
}
else
{   printf("yellow");
}
printf("orange");
```

ip: a is 10, b is 20
op: ?

ip: a is 10, b is 30
op: ?

ip: a is 20, b is 20
op: ?

5)
```
if( a==10 )
{       if( b==20 )
        {    printf("green");
        }
}
else
{       printf("white");
}
```
ip: a is 10, b is 20
op: ?

ip: a is 10, b is 30
op: ?

ip: a is 20, b is 20
op: ?

6)
```
if( A==10 && B==20 )
{       printf("hello");
}
else
{       printf("world");
}
```
ip: A=10, B=20
op: ?

ip: A=10, B=30
op: ?

7)
```
if( A==10 || B==20 )
{       printf("hello");
}
else
{       printf("world");
}
```
ip: A=10, B=20
op: ?

ip: A=20, B=20
op: ?

ip: A=20, B=30
op: ?

8)
```
if( A==10 && B==20 || C==30 )
{       printf("hello");
}
else
{       printf("world");
}
```
ip: A=10, B=20, C=40
op: ?

ip: A=10, B=10, C=30
op: ?

ip: A=5, B=20, C=30
op: ?

ip: A=20, B=20, C=20
op: ?

9)
```
if( A==10 &&  (B==20 || C==30) )
{       printf("hello");
}
else
{       printf("world");
}
```
ip: A=10, B=20, C=40
op: ?

ip: A=10, B=10, C=30
op: ?

ip: A=5, B=20, C=30
op: ?

ip: A=10, B=20, C=20
op: ?

10)
```
if ( (A==5 ||  B==5 || C==5 ) &&  D>10  )
        printf("yes");
    else
        printf("no");
```
ip: A=5, B=6, C=7, D=20 ?
ip: A=6, B=7, C=8, D=20 ?
ip: A=5, B=7, C=8, D=5 ?

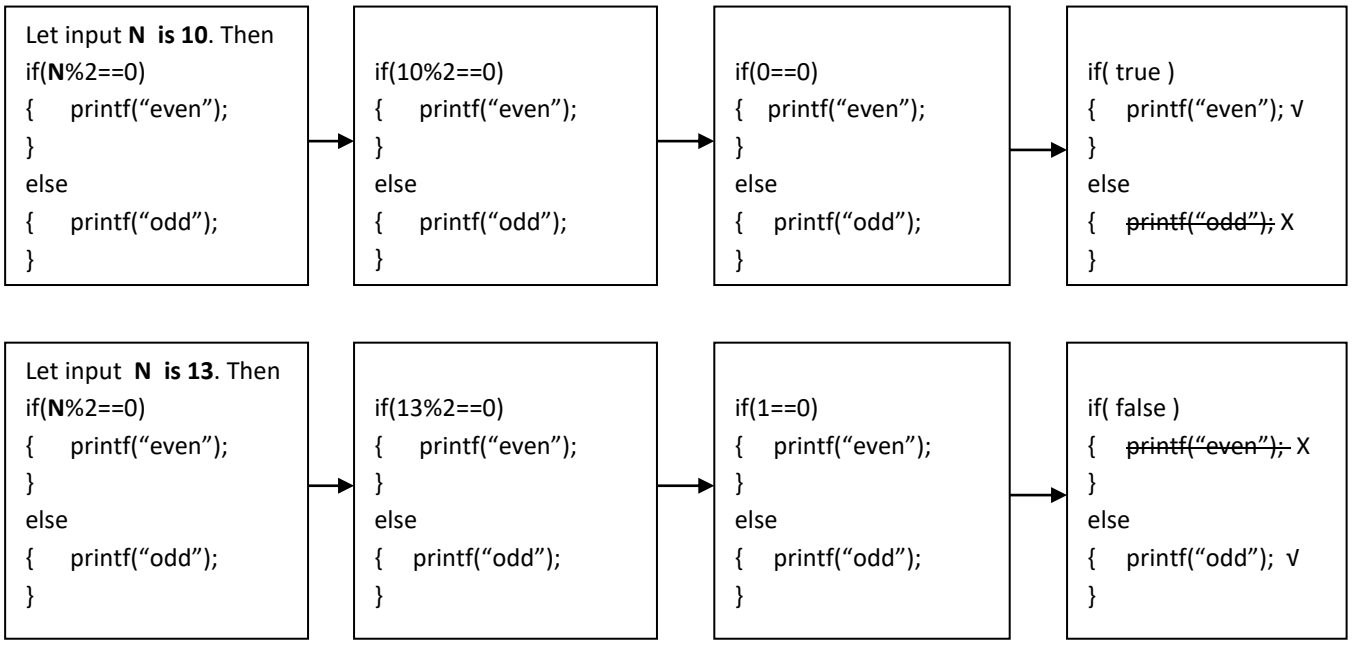
---

11)
```
if(  0  ==  10%5 )
        printf(" Good ");
  else
        printf("very Good ");
```
op : ?

12) Demo: Finding whether a given number is odd or even. This shows how to divide and check the remainder. The operator % gives the remainder of a division. For example, 7 % 5 → 2, 12 % 2 → 0 (whereas 12 / 2 → 6). If a number N is even, it divides perfectly by 2, which means the remainder of the division should be zero. We know that numbers like 2, 4, 6, 8, etc., are even. In a C program, the logic is

```
if( N%2==0 )
{    printf("even");
}
else
{   printf("odd");
}
```

the evaluation of if-statement is as follows



---

13) Demo, this program shows how to accept single digit from KB and printing in English words.

ip: 3 ip: 5 ip: 23
op: three op: five op: error, input is not a digit

```
int n;
char  *x;          // to store characters(string)
printf("enter single digit :");
scanf("%d", &n);

if( n==0 )
{   x="zero";
}

if( n==1 )
{   x="one";
}
----
----
if(n==9)
{   x="nine";
}

if( n>9 )
{   x="error, input is not a digit";
}
printf("output  is %s",  x);       //  %s is used for strings
```

---

14) If an integer N is input through the keyboard, write a program to print its absolute value [mod(N)].
The input number may be positive or negative, but the output should always be positive.
Logic: If the input is negative, convert it to positive by multiplying it with -1. **if N<0 then N is said to be -ve**

ip: –12 ip: 12 ip: -19
op: 12 op: 12 op: 19

step1: take N as variable
step2: scan N as input value from KB
step3: if N is –ve then **// if(N<0)**
{
here change -N to +N by multiplying it with -1
}
step4: print N as output value.

---

15) If two integers (A, B) are input through the keyboard, write a program to find the difference value (A-B).
The output must be printed as a positive. If the result of A-B is negative, convert it to positive like above.

ip: 12 18 ip: 18 12
op: 6 op: 6

Method1:

step1: take variable names as: A, B, X
A, B is to hold input values, and 'X' is to hold output value.
step2: scan two values into A , B
step3: X=A-B // here result of 'X' might be -ve
step4: if 'X' is –ve then
{ here change -X to +X
}
step5: print 'X' as +ve output

Method2:

step1: take variable names as: A, B, X
step2: scan two values into A, B
step3: if A>B then
{ store A-B value to X
}
else
{ store B-A value to X
}
step4: print 'X' as +ve output

note: method1 is better than method2 in terms of machine code generation and execution time.
Try in both methods

---

16) Accept a value(N) from keyboard and print output as red/green/yellow color.
if N is –ve then print "red color", if N is +ve then print "green color", if N is zero then print "yellow color"

ip: -12 ip: 14 ip: 0
op: red color op: green color op: yellow color

```
eg:  char *x;
     int N;
     ----
     if(N<0)
     {  x="red color";        // check 13th problem
     }
     ----
     printf(" output is: %s", x );
```

---

17) if two integers(A,B) are input through the keyboard, write a program to find biggest among them.

ip: 12 45 ip: 15 53

op: 45 op: 53

procedure

step1: Take variable names as: A, B, X

A,B is to hold input values, and X is to hold output value.

step2: scan two values into A,B

step3: if A>B then

{ store A to X as big

}

else

{ store B to X as big

}

step4: now X is holding big value of (A,B), so print 'X' on the screen

---

18) If three integers(A, B, C) are input from KB, code to replace 3 values into sorted order(A<B<C)

ip: 12 5 65 ip: 12 6 3 ip:10 5 2

op: 5 12 65 op: 3 6 12 op: 2 5 10

step1: take variable names as: A, B, C, temp; // temp is an extra variable needed while swapping values

step2: scan 3 values into A, B, C // let A,B,C are 10 5 2

step3: if(A>B) then

{ swap A,B (using temp) // this swaps 10 5 2 → 5 10 2

}

if(A>C) then

{ swap A,C // this swaps 5 10 2 → 2 10 5

}

if(B>C) then

{ swap B,C // this swaps 2 10 5 → 2 5 10

}

step4: print A, B, C // output will be sorted order, this logic works for any kind of input values.

---

19) leap year is a year which is divisible by 4 , write a program to check given year is leap-year or not ?

ip: 2004 ip: 2003

op: leap-year op: non leap-year

// divisible by 4 means, the remainder of division is equal to zero. (use % operator for remainder)

---

20) The ABC cloth show room offering discount as given below

if billed amount<=3000 then discount is zero,

if above>3000 then discount is 25% on billed amount

finally print discount.

input: billed amount

output: discount

note: take variable names as : billedAmount, discount (float type)

---

21) If basic salary of employee scanned through the keyboard, find net salary as given below

If basic salary <= 20000 then

DA is 5% on basic salary // DA=basicSalary*5/100 ; [DA means Dearness Allowance]

TA is 7% on basic salary // TA=basicSalary*7/100; [TA means Travelling Allowance]

If basic salary > 20000 then (we can use 'else')

DA is 9% on basic salary

TA is 11% on basic salary

HRA (House Rent Allowance) is 24%, this is common for both below & above 20000 salaried employees.

Now find net salary as sum of all allowances [net salary = basicSalary + HRA + DA + TA]

note: take variable names as: BS, NS, HRA, DA,TA (take as float type)

---

22) Write a program to accept student marks(M) of one subject and print result pass/fail.

College adding 10 grace marks to all students, after adding, it should not cross 100. (max marks is 100)

if Ramu got 60 marks then his marks is 70, if he got 93 then his marks 100.

Finally print pass/fail. if M<50 then print "fail" or else "pass" and also print final marks.

step1: scan M as marks (M)

step2: add 10 marks to M

step3: if M crosses 100 then set to 100. // this is like: if(M>100) { M=100; }

step4: if M < 50 then say "failed" or else say "passed"

---

23) for children, government providing nearly ticket free travelling for hill station by rope.

- the ticket cost for children <=16 years is 10/- rupees (fixed).
- for those age>16 then ticket cost depends upon weight of person, the cost is 2/- per every 5 kilograms. Now write a program to scan age of person, if age<=16 then take cost as 10/- or else scan weight of a person and calculate cost as (weight/5)*2/-

procedure:

step1: take variable names as: age, weight, cost (int-type)

step2: scan age of a person

step3: if age<=16 then

```
{   cost is 10 Rupees      // cost=10;
}
else                       // age >16
{   here scan weight of a person
    cost is weight/5*2
}
```

step4: print cost as output.

About Pair of Braces { }

The pair of braces is optional when if-body or any control statement body contains only single instruction.
That is, the pair of braces is not required when if-body or else-body contained only single instruction.
The following code shows how to remove braces when single instruction exists in if-body or else-body.

```
if( A>B )
{      printf("Red");
}
printf("Blue");
printf("Green");
-------
```

→

```
// code after removing unnecessary braces
if( A>B )
      printf("Red");
printf("Blue");
printf("Green");
-------
```

```
if( A>B )
{      printf("Red");
}
else
{      printf("Blue");
}
printf("Green");
-------
```

→

```
// code after removing unnecessary braces
if( A>B )
      printf("Red");
else
      printf("Blue");
printf("Green");
-------
```

```
if( A>B )
{      printf("Red");
}
else
{      printf("Blue");
       printf("Green");
}
printf("White");
-------
```

→

```
// code after removing unnecessary braces
if( A>B )
      printf("Red");
else
{      printf("Blue");
       printf("Green");
}
printf("White");
-------
```

```
if( A>B )
{      printf("Red");
       printf("Blue");
}
else
{      printf("Green");
}
printf("White");
-------
```

→

```
// code after removing unnecessary braces
if( A>B )
{      printf("Red");
       printf("Blue");
}
else
       printf("Green");
printf("White");
-------
```

```
if(A>0)
{     printf(" A is +VE");
}
if(A<0)
{     printf(" A is -VE");
}
if(A==0)
{     printf(" A is Zero");
      printf(" A is +VE");
}
```

→

```
// code after removing unnecessary braces
f(A>0)
    printf(" A is +VE");
if(A<0)
      printf(" A is -VE");
if(A==0)
{     printf(" A is Zero");
      printf(" A is +VE");
}
```

Code with Indentation

the instructions inside if-block or else-block or any block should be started with indentation.
The indentation means giving tab-space before instructions (it is like starting space before paragraph begins)
This **indentation** makes the program easy to read and understand. It signifies which instruction is inside and which is not.
Remember if we do not follow indentation then code becomes difficult to read & understand.
C compiler does not force you to follow it, so it does not show any error if indentation is ignored.
(Python language forces you to follow it). Following example shows indentation.

if(A>B) printf("Red"); → **with indentation** printf("Blue"); printf("Green"); ------	if(A>B) printf("Red"); → **without indentation (makes confusion)** printf("Blue"); printf("Green"); -------
if(A>B) printf("Red"); → **with indentation** else printf("Blue"); → **with indentation** printf("Green"); ------	if(A>B) printf("Red"); → **without indentation (makes confusion)** else printf("Blue"); → **without indentation (makes confusion)** printf("Green"); ------
if(A>B) { printf("Red"); → **with indentation** printf("Blue"); } printf("Green"); ------	if(A>B) { printf("Red"); → **without indentation (little confusion)** printf("Blue"); } printf("Green");
with indentation, here no confusion if(a==10) { printf("red"); if(b==20) printf("green"); else { printf("blue"); printf("white"); } printf("black"); }	**Code without indentation, here lot of confusion** if(a==10) { printf("red"); if(b==20) printf("green"); else { printf("blue"); printf("white"); } printf("black"); }
Many professionals used to write if-else in single line as: if(A>B) X=A; else X=B; this is actual syntax provided by the C-Professionals. But this syntax is not suitable for beginners.	**Example2: if if-body has single instruction then write in the same line of if-statement, this is as shown below** if(A>0) printf(" A is +VE"); if(A<0) printf(" A is -VE"); if(A==0) { printf(" A is Zero"); printf(" A is +VE"); }

Count and Boolean logics

sometimes, the count logic makes the program easy, here we check and count the input values in favor of +ve or -ve side and we take final decision on count. Let us see following examples.

Note: this logic explained without using logical operators (&& , ||)

Demo1: Let student has 2 subjects, If he obtained>50 in 2 subjects then print "passed" or else "failed".

ip: 51 60 ip: 30 60 ip: 90 60
op: passed op: failed op: passed

```
logic is:  step1:  first take 'count' as zero   // count=0;
           step2:   scan( m1 , m2 )
           step3:   if( m1 > 50 )
                        count++;
           step4:   if( m2 > 50 )
                        count++;
           step5:  if( count==2 )  printf("passed");
                   else   printf("failed");
```

Demo2: Let there are 3 values A, B, C, now finding at least one value is -ve or not?

ip: 10 20 -30 ip: 10 -20 -30 ip: 10 20 30
op: -ve found op: -ve found op: -ve not found

```
logic is:   step1:  first take 'count' as zero      // count=0;
            step2:  scan( A,B,C )
            step3:  if( A < 0 )
                        count++;
            step4:  if( B < 0)
                        count++;
            step5:  if( C < 0)
                        count++;
            step6:  if( count==0)     printf("-ve not found ");
                    else   printf("-ve found");
```

---

24) code to check given input 'time' values are valid or not ? (take time as 3 values for H,M,S)
if seconds >59 or minutes >59 or hours > 12 then invalid time

ip: 10 4 30 ip: 15 4 30 ip: 5 44 30 ip: 3 4 89
op: valid inputs op: invalid inputs op: valid inputs op: invalid inputs

---

25) code to scan 3 values from keyboard and count how many +ve and –ve values exist.

ip: 10 -14 30 ip: -15 -4 -19 ip: 3 4 89
op: + ve count is: 2 op: + ve count is: 0 op: + ve count is:3
- ve count is: 1 - ve count is: 3 - ve count is:0

Boolean/bool logic

What is bool or boolean: A well-known Scientist **George Boole** applied probability theory to solve some kind of mathematical algebraic problems (Boolean Algebra). We know, in math, probability value ranges 0 to 1.
Here the value 0 represents no-hope, whereas, 1 represents 100% hope. (0.5 represents 50% hope).
These values adapted to computer science and calling them as Boolean values, used for 2-way decisions output problems. The problems like pass/fail, exist/not-exist, even/odd, prime/not-prime, etc.
We can use any other values instead of 1/0, but it is informal.
Generally, 0 is used for –ve result like failed case, whereas 1 is used for +ve result like success case.

Procedure for Boolean logic:
step1: take Boolean variable mostly with name 'bool'
step2: first, we need to take assumption of result +ve or –ve side (most of the time it is +ve side)
so take boo1=1;
step3: now check each input value in opposite side of assumption, check all values one by one.
if any input value is invalid then set bool to 0.
step4: finally, the result in bool in the form of 1 or 0. Now check bool and print result.

Let us see following examples
Demo1) If marks of 2 subjects scanned from KB, write a program to print student is passed or failed;
If student obtained>=50 in 2 subjects then print "passed" or else "failed".

ip: 51 60	ip: 30 60	ip: 90 60
op: passed	op: failed	op: passed

```
int bool=1;        // let us assume student has passed, so take 'bool' as 1. (we can take any name for bool).

if(m1<50)          // now checking opposite side of above assumption
    bool=0;        // if this condition is true, the student failed, so set 'bool' to 0

if(m2<50)
    bool=0;    // if this condition is true, the student failed, so set 'bool' to 0
```

/* in the above code, if at least one if-condition is true then 'bool' becomes 0, it indicates that, the student has failed. now result is in 'bool' in the form of 1 or 0, so check 'bool' value and print "pass" or "fail" */

```
if(bool==1)  printf("passed");
if(bool==0)  printf("failed");
```

Demo2) finding given input number(N) is in b/w 10 to 20 or not? (finding using bool logic)

ip: 12	ip: 25
op: yes, in limits	op: not in limits

```
k=1;          // here 'k' is Boolean variable, and assume N is in b/w 10 to 20, so taking k=1
if( N<10)
    k=0;      // here N is below 10, so not in 10 to 20, then taking k=0

if( N>20)
    k=0;      // here N is above 20, so not in 10 to 20, then taking k=0

if( K==1)  printf("yes, it is in b/w 10 to 20");
else   printf("no, it is not in b/w 10 to 20");
```

---

26) if three integers are input through the keyboard, then find at least one value is –ve or not?
try using Boolean logic (do not use logical operators && , ||)

ip: 10 20 -30 ip: 10 -20 -30 ip: 10 20 30
op: -ve exist op: -ve exist op: -ve not exist

---

27) check given input time (h, m, s) values are valid or not ? use Boolean logic.
if hours > 12 or minutes > 59 or seconds > 59 then wrong input values.

ip: 10 4 30 ip: 15 4 30 ip: 5 44 30
op: valid inputs op: invalid inputs op: valid inputs

Programs using Logical operators (&& , ||)

---

28) If marks of 2 subjects scanned from keyboard, write a program to print student is passed or failed;
if student obtained>=50 in 2 subjects then print "passed" or else "failed".

ip: 51 60 ip: 30 60 ip: 90 60
op: passed op: failed op: passed

A) find using logical operator AND (&&)
for example: if (m1>=50 && m2>=50) printf("passed") else printf("failed");

B) find using logical operator OR (||)
for example: if (m1<50 || m2<50) printf("failed") else printf("passed");

---

29) Rewrite above program by taking 3 subject marks.

A) find using logical operator AND (&&)
for example: if (m1>=50 && m2>=50 && m3>=50) printf("passed") else printf("failed");

B) find using logical operator OR (||)

---

30) Code to find given input number(N) is in between 10 to 20 or not?

ip: 12 ip: 25
op: yes, it is op: no, it is not

A) find using logical operator && (AND operator)
B) find using logical operator || (OR operator)

---

31) if three integers are input through the keyboard, then find at least one value is –ve or not?

ip: -12 34 -42 ip: 52 64 -72 ip: 62 44 42
op: "yes, –ve exist" op: "yes, –ve exist" op: "–ve is not exist"

A) find using logical operator || (OR operator)
B) find using logical operator && (AND operator)

---

32) Scan the time values (Hours, Minutes, Seconds) from keyboard and find valid values or not?
if hours > 12 or minutes > 59 or seconds > 59 then wrong input values.

ip: 15 95 12 ip: 2 16 12 ip: 13 91 92
op: "invalid inputs" op: "valid inputs" op: "invalid inputs"

A) find using logical operator || (OR operator)
B) find using logical operator && (AND operator)

Nested-if construct

Constructing one **"if-statement" within another "if-statement"** is said to be nested-if. For example,

```
if( condition1 )            // this is said to be "outer-if"
{
    if( condition2 )        // this is said to be "inner-if"
    {   ------
        ------
    }
}
```

note: we can nested as many times as we want and every if-statement may have its own else-part.

for example

```
if( age<10 )                                // outer-if
{
    if( weight<30)                          // inner-if
        printf("very good weight");
    else
        printf("overweight");
}
else
{
    if( weight<70)                          // inner-if
        printf("very good weight");
    else
        printf("overweight");
}
```

---

33) Demo program finding big of 3 values using nested-if style

// simple nested-if style

```
if( A>B )
{   if( A>C )
        X=A;
    else
        X=C;
}
else
{   if( B>C )
        X=B;
    else
        X=C;
}
printf("big is %d", X );
```

// nested-if with logical operators

```
if( A>B  && A>C )
    X=A;
else
{   if( B>C )
        X=B;
    else
        X=C;
}
printf("big is %d", X );

// in this way, by adding logical operators to
 nested-if, we can simplify the code.
```

For nested-if statements, there is no specific syntax—they can take any form. They can be written in any permutation or combination, depending on the problem, and can be nested as many times as needed. Each outer or inner if-statement may or may not have its own else-part.

Many problems involve alternative decisions, such as finding the biggest of four numbers. In such cases, each else-part is nested. First, we assume A is the largest; we check A, if not, we check B, then C, and finally D. In this way, many problems are structured. The following examples may help you understand how to write such alternative decisions."

34) Demo program finding biggest of 4 numbers. Let A,B,C,D are 4 numbers
1) writing code using 4 independent **if-statements** 2) using **nested-if** style

// using 3 independent if-statements

```
if( A>B && A>C && A>D)
{   X=A;
}

if( B>A && B>C && B>D )
{   X=B;
}

if( C>A && C>B && C>D)
{   X=C;
}

if( D>A && D>B && D>C)
{   X=C;
}
```

Note: even if first condition is true , the computer unnecessary checks all the remaining.

// nested-if style (simplified solution)

```
if( A>B && A>C && A>D )
        X=A;
else
{       if( B>C && B>D )
                X=B;
        else
        {       if(C>D)
                        X=C;
                else
                        X=D;
        }
}
```

here, if one condition is true, then rest of the conditions are by passed(not checked). So this is best logic.

35) Demo code, finding age group of a person as given below
if age<=12 then say child ,
if age>12 and age<=19 then say teenager ,
if age>19 and age<=50 then say younger ,
if age>50 then say old-aged .

// simple-if style

```
if( age <= 12 )
      x="child";

if( age>12 && age<=19 )
      x="teenager";

if( age>19 && age<=50 )
      x="young";

if(age>50 )
      x="teenager";

printf("output is  %s " , x);
```

// nested-if style

```
if( age <=12 )
        x="child";
else
{       if( age<=19 )
                x="teenager";
        else
        {       if( age<=50 )
                        x="young";
                else
                        x="old";
        }
}
printf("output is  %s " , x);
```

36) Complete the code to find how many digits exist in a given N. Assume N lies in between 0 to 99999

ip: 234 ip: 3456 ip: 12234 ip: 3

op: 3 op: 4 op: 5 op: 1

```
if( N<10 )
        count=1;

if( N>=10 &&  N<100 )
        count=2;
-------
-------
```

→

```
if( N<10 )
      count=1;
else
{     if( N<100 )
            count=2;
--------
--------
```

Note: complete both styles

---

37) Program to accept salary from keyboard and find tax

if salary<=10000 then tax is zero

if salary>10000 and salary <=20000 then tax is 5% on salary

if salary>20000 and salary <=30000 then tax is 8% on salary.

if salary>30000 then tax is 12% on salary.

---

38) The C-Family library charges a fine for every late returned book.

For first 10 days, the fine is 5.00/-

For 11-20 days, the fine is 12.00/-

For 21-30 days, the fine is 28.00/-

For above 30days, the fine is 1/- per a day (if late is 34 days then fine is also 34.00/-)

```
// eg:  if(daysLate <=10)
            fineAmount=5;
```

ip: 16 ip: 45 ip: 6 ip: 22

op: 12rs op: 45rs op: 5rs op: 28rs

Note: input is no.of days late and output is fine. Take variable names as: daysLate , fineAmount.

---

39) If the number of units scanned from keyboard, find electricity bill as given below tariff

tariff 1: if units <= 100

bill is 3.50/- per unit

tariff 2: if units>100 and units<=200

upto 100 units as above said(3.50/-), For remaining units(units-100), charge is 5.00/-

tariff 3: if units>200

upto 200 units as above said, For remaining units(units-200), charge is 8.00/- per unit

ip: units: 78 ip: units: 123 ip:225

op: bill=78*3.50→273 op: 100*3.50+(123-100)*5.00→465 op:100*3.50+100*5.00+(225-2001)*8.00

---

40) If two dates are input from keyboard, write a program to find latest date among them.
Take input dates as 6 values (d1,m1,y1 as date1) and (d2,m2,y2 as date2)

ip: 29 2 2012
30 12 2010
op: date-1 is latest

```
if(y1>y2)
    print date-1 as latest
else
{   if(y1<y2)
        print date-2 as latest
    else
    {   the control comes to this block when above two condition are false.
        if above two conditions are false means, the years are said to be equal.
        so here compare months & days like above .
        ------
```

---

41) If marks of 3 subjects of a student are input through keyboard and find result

- if student obtained <35 in any subject then print "failed" **// if(m1<35 || m2<35 || m3<35) …**

Otherwise print "A-grade/B-grade/C-grade" based on average.

- if average >= 60 then print "passed in A-grade"
- if average 50 to 60 then print "passed in B-grade"
- if average <50 then print "passed in C-grade"

ip: 80 90 90	ip: 45 60 50	ip: 40 36 41	ip: 20 40 50
op: passed in A-Grade	op: passed in B-Grade	op: passed in C-Grade	op: Failed

---

42) If marks of 2 subjects are input through the keyboard, write a program to print result.
logic: Generally, to get pass mark, student must obtain >= 50 in 2 subjects, however, the university has given an exemption: if a student scores **10 marks less** in **any one subject**, they are still considered **passed**.
That means the student must get **at least 50 marks in one subject and at least 40 marks in the other**.

ip: 70 46	ip: 77 66	ip: 45 45	ip: 46 59	ip: 70 74
op: passed	op: passed	op: failed	op: passed	op: passed

---

43) To pass the exam, a student must score **at least 50 marks in a subject**. The student is given **two attempts**.
If the student scores **50 or more in the first attempt**, they are **declared passed**.
If the student scores **less than 50 in the first attempt**, they are allowed to take a **second attempt** to pass or fail.
Logic: If he got <50 in first attempt then again scan marks as second attempt and print result pass or fail.

---

44) if 3 subject marks are scanned from KB then print how many subjects found above 90 marks

ip: 42 94 91	ip: 99 94 92	ip: 62 44 42
op: 2 subjects>=90	op: 3 subjects>=90	op: 0 subjects>=90

note: try using logical operators, and also try using count logic.

---

45) Accept 3 values from KB, input values may be +ve/-ve, but find sum of +ve values only.

ip: -2 3 4	ip: -2 -6 -4	ip: -5 45 14
op: sum of +ve: 3+4 → 7	op: sum of +ve: 0	op: sum of +ve: 59

About braces: observe if-block have 2 statements in the following example

46) complete the code to find age group and normal weight of a person as given below

if age<=12 then say child and normal weight is 20

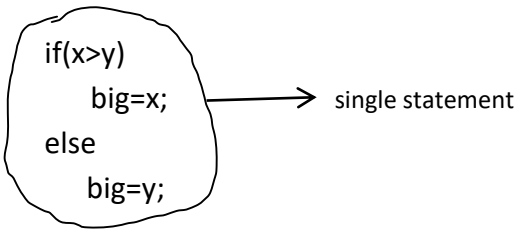
if age>12 and age<=19 then say teenager and normal weight is 40

if age>19 and age<=50) then say younger and normal weight is 60

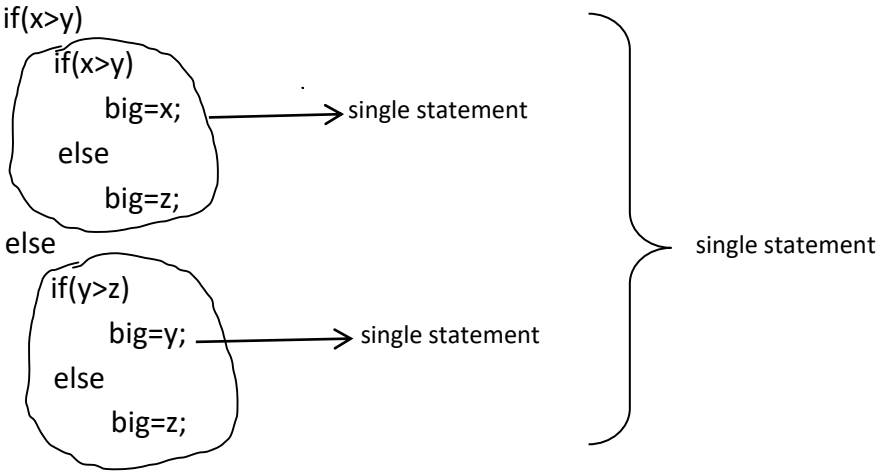
```
if( age<=12)
{       X="child";      // braces needed because two instructions exist here.
        weight=20;
}
else
{       if(age<=19)
        {       X="teenager";
                weight=40;
        }
        ----
        ----
}
printf(" age is  %s , normal weight is %d " ,  X, weight );
```

More about Pair of Braces { }

*** As per C language syntax, if-else control statement can be taken as single-compound-statement even though they seem to be two, for example, the following if-else can be taken as single.**



*** As inner if-else statement is single, braces are not required to make into single for outer-if.**
Based on this theory, we can remove braces for outer-if also. Let us see following example



In this way, in C, any control statement can be taken as single statement including nested-if,
thus above entire if-statement can be taken as single-compound-statement.

But sometimes, some people used to give braces for clarity purpose even though they are not required.

Example-3

```
if(x>y  && x>y )
      big=x;
else
      if(y>z)
            big=y;
      else
            big=z;
```

single statement (the inner if-else)

single statement (the outer if-else)

Example-4

```
if( a>0 )
{     printf("Hello1");
      printf("Hello2");
}
else
{     printf("Hello3");
      printf("Hello4");
}
```

single statement

47) Demo program finding biggest of 4 numbers. Let A,B,C,D are 4 numbers

with braces

```
if( A>B && A>C && A>D )
      X=A;
else
{     if( B>C && B>D )
            X=B;
      else
      {     if(C>D)
                  X=C;
            else
                  X=D;
      }
}
```

without braces

```
if( A>B && A>C && A>D )
      X=A;
else
      if( B>C && B>D )
            X=B;
      else
            if(C>D)
                  X=C;
            else
                  X=D;
```

with braces

```
if( age <=12 )
      x="child";
else
{     if( age<=19 )
            x="teenager";
      else
      {     if( age<=50 )
                  x="young";
            else
                  x="old";
      }
}
printf("output is  %s " , x);
```

without braces

```
if( age <=12 )
      x="child";
else
      if( age<=19 )
            x="teenager";
      else
            if( age<=50 )
                  x="young";
            else
                  x="old";

printf("output is  %s " , x);
```

if-else-if ladder style

This is not a special control statement; it is one kind of written style of **nested-if** when several alternative decisions exist. Normal nested style leads to heavy indentation when more alternatives exist.
So it is nothing but rearrangement of **nested-if statement** like a straight-line unlike crossed-line as shown in the below syntax.
Here we remove unnecessary braces and we write all nested-if statements in the same line of else-block then automatically form as ladder style.

This is applied, when needed to process one decision from several alternative decisions. This structure is also called '**if-else-if**' staircase because of its appearance. The main advantage of this structure is, easy to code, understand, and debug. Let us see one example

50) Above Programs using if-else-if ladder Style

If student obtained <35 in any one of 3 subjects then student is "failed" or else "passed"

Nested-if style

```
if( A<35)
        printf("failed");
else
{       if( B<35)
              printf("failed");
        else
        {       if( C<35)
                      printf("failed");
              else
                      printf("passed");
        }
}
```

ladder style

```
if( A<35)
        printf("failed");
else  if( B<35)
        printf("failed");
else  if( C<35)
        printf("failed");
else
        printf("passed");
```

51) demo program for finding biggest of 4 numbers in if-else-if ladder style

Normal nested-if

```
if( a>b && a>c && a>d )
        x=a;
else
{       if(b>c &&  b>d )
              x=b;
        else
        {       if(c>d)
                      x=c;
              else
                      x=d;
        }
}
printf("\n biggest=%d", x )
```

if-else-if ladder style

```
if( a>b && a>c && a>d )
        x=a;
else if( b>c &&  b>d )
        x=b;
else if( c>d )
        x=c;
else
        x=d;
printf(" \n biggest=%d",  x );
```

52) Demo program showing age group and general weight of a person

if age<=12 then say child and general weight is 20

if age>12 and age<=19 then say teenager and general weight is 40

if age>19 and age<=50) then say younger and general weight is 60

if age>50 then say old-age and general weight is 80

```
 if(age<13)
{       X="child";
        weight=20;
}
else
{       if(age<20)
        {       X="teenager";
                weight=40;
        }
        else
        {       if(age<51)
                {       X="younger");
                        weight=60;
                }
                else
                {       X="old";
                        weight=80;
                }
        }
}
```

→

```
if(age<13)
{       X="child";
        weight=20;
}
else   if(age<20)
{       X="teenager";
        weight=40;
}
else  if(age<51)
{       X="younger";
        weight=60;
}
else
{       X="old";
        weight=80;
}
```

---

53) Demo program to accept salary from keyboard and find tax

if salary<=10000 then tax is zero

if salary>10000 and <=20000 then tax is 5% on salary

if salary>20000 then tax is 8% on salary.

ip: enter salary: 9000 ip: enter salary: 20000 ip: enter salary:42000

op: tax = 0/- op: tax = 1000/- op: tax = 3,360/-

Normal if-statement	Nested-if style	Ladder style
if(salary<=10000) tax=0; if(salary>10000 && salary <=20000) tax=0; if(salary<=10000) tax=0; printf("tax is %f", tax);	if(salary<=10000) tax=0; else { if(salary <=20000) tax=salary*5/100; else tax=salary*8/100; } printf("tax is %f", tax);	if(salary<=10000) tax=0; else if(salary <=20000) tax=salary*5/100; else tax=salary*8/100; printf("tax is %f", tax);

→ →

---

54) The C-Family library charges a fine for every book late returned. (try using ladder style)

For first 10 days, the fine is 5.00/-

For 11-20 days, the fine is 12.00/-

For 21-30 days, the fine is 28.00/-

For above 30days, the fine is 1/- per a day (if 34 days late then fine is also 34.00/-)

ip: 16 ip: 45 ip: 6 ip: 22

op: 12rs op: 45rs op: 5rs op: 28rs

Note: input of this program is no.of days late and output is fine amount.

Take variable names as: daysLate, fineAmount.

---

55) Find how many digits exist. Let us assume input value of N lies in between 0 to 99999

ip: 234 ip: 3456 ip: 12234 ip: 3

op: 3 op: 4 op: 5 op: 1

---

56) code to check given triangle is equilateral(all sides 60^), isosceles(two sides equal), scalene (all diff).

note: sum of 3 angles should be 180 before checking, if not then show an error message invalid input.

ip: 100 80 40 ip: 50 100 30 ip: 60 60 60 ip: 50 50 80

op: invalid input op: scalene op: equilateral op: isosceles

procedure:

step1**:** if(a+b+c != 180)

printf("invalid input");

else …… (don't print valid, here print equilateral/isosceles/scalene using following steps)

step2: if three are equal then print equilateral // if(a==b && b==c)

step3: if any two are equal then print isosceles // if(a==b || b==c || c==a)

step4: if above two conditions are failed, it means, all are different, so print as scalene

---

57) Write a program to check whether given triangle is (let input values are valid)

case1) equilateral

case2) isosceles with right angle (if any one side is 90 and any two are equal)

case3) isosceles

case4) scalene with right angle.

case5) scalene

ip: 50 100 30 ip: 90 30 60 ip: 60 60 60 ip: 50 50 80 ip: 45 90 45

op: scalene op: scalene+right angle op: equilateral op: isosceles op: isosceles+right angle

---

58) If date(month, year) is input through KB, write a program to print how many days exist in that month.

- February with leap year has 29 days, eg: if(month==2 && year%4==0) days=29;
- February with non- leap year has 28 days
- the months such as 4, 6, 9, 11 have 30-days (April, June,….etc)
- the months such as 1, 3, 5, … have 31-days (Jan, Mar,….etc)

ip: 2 2010 ip: 2 2000 ip: 4 2000 ip: 5 2001

op: 28 days ip: 29 days ip: 30 days ip: 31 days

---

59) If date(d,m,y) is input through the keyboard, write a program to find whether it is valid date or not?

ip: 30-2-2010 ip: 31-4-2000 ip: 30-4-2000
op: invalid date op: invalid date op: valid date

step1: scan date as (d,m,y)
step2: Let us assume, date is valid, so take bool=1
step3: if(m<1 || m>12 || d<1 || d>31) // if anyone is true then date is invalid, so replace bool with 0
bool=0;
else if(month is February and leap-year and d>29) // if(m==2 && y%4==0 && d>29)
bool=0;
else if(month is February and non-leap-year and d>28)
bool=0;
else if((month is 4 or 6 or 9 or 11) and d>30)
bool=0;
step4: // now the result status is in bool variable in the form of 1 or 0, so check and print result
if(bool is 1) then print("Given date is valid");
if(bool is 0) then print(" Given date is **in**valid");

---

60) if valid date (d, m, y) is input through keyboard, write a program to increment it by one day

ip: 29-2-2012 ip: 31-12-2012 ip: 28-2-2010 ip: 2-2-2012
op: 1-3-2012 op: 1-1-2013 op: 1-3-2012 op: 3-2-2012

step1: scan date as d, m, y
step2: increment day, this is like d++ // because we have to increment date by 1-day
step3: after incrementing day, if day crossed month ending limits then shift to next month, for example
case1: if crossed February month ending : if(m==2 && y%4==0 && d>29) then d=1, m++; (leap-year)
case2: if crossed February month ending : if(m==2 && y%4!=0 && d>28) then d=1, m++; (non-leap)
…
stepX: finally print incremented date

---

61) Write a program to display the type of the roots of a quadratic equation (ax^2+bx+c) given by its coefficients say a, b and c. Here a, b and c are input numbers.

Process: To know the type of roots of an equation, first of all, evaluate the value of (b^2-4ac), let it is 't'

if t < 0 then
print output as "roots are imaginary"
if t > 0 then
root1 is (-b+sqrt(t))/(2*a) // **note:** here sqrt() is a library function, so add #include<math.h>
root2 is (-b-sqrt(t))/(2*a)
print root1, root2 values
if t == 0 then
root1 = root2 = -b/(2*a)
print "two roots are equal" and also print root1, root2 values

ip: enter a, b, c values: 2 4 2 (equation is: $2x^2 + 4x + 2$)
op: two roots are equal and values are: root1= -1.00 , root2= -1.00

ip: enter a, b, c values: 2 3 4 (equation is: $2x^2 + 3x + 4$)
op: roots are imaginary

ip: enter a, b, c values: 2 8 3 (equation is: $2x^2 + 8x + 3$)
op: root1= -0.42 , root2= -3.58

Following example finds student is passed or not? This program explained with many logics.
The student has 3 subjects named A, B, C. If he got>50 in all subjects then he is passed, otherwise failed.

using logical operator AND (&&)

```
if( A>50  &&  B>50 && C>50 )
    x="passed";
else
    x="failed";
```

using logical operator OR (||)

```
if( A<50  ||  B<50 || C<50 )
        x="failed";
else
        x="passed";
```

using nested-if without using logical operators

```
if( A>50 )
        if( B>50 )
             if( C>50 )
                  x="passed";
             else
                  x="failed";
        else
             x="failed";
else
      x="failed;
```

if-else-if ladder style without logical operators

```
if( A < 50 )
        x="failed";
else if( B < 50 )
         x="failed";
else if( C < 50 )
        x="failed";
else
        x="passed";
```

count logic, without nested or logical operators

```
count=0;
if ( A<50 )  count++;
If ( B<50 )  count++;
If ( C<50 )  count++;
If ( count ==0 )
    printf("passed");
else
    printf("failed");
```

boolean logic, without nested or logical operators

```
bool=1;
if ( A<50 )  bool=0;
If ( B<50 )  bool=0;
If ( C<50 )  bool=0;
If ( bool==1 )
    printf("passed");
else
    printf("failed");
```

Home work

---

70) Write a program to accept 4 values from keyboard and print biggest.

step1) Let us take four variables A,B,C,D for input, also take X to store output value.

step 2) Let us assume 'A' is big, so store directly 'A' value to X (eg: X=A)

step 3) Now compare X with B, if B is bigger than X, then store B value to X (replaces A by B in X)

step 4) Now compare X with C, if C is bigger than X, then store C value to X

step 5) Now compare X with D, if D is bigger than X, then store D value to X

step 6) Finally, the big value in X, so print it

---

71) Write a program to scan '**N'** as input, and print denomination of it (the denomination values are 5, 2, 1)

ip: 18 ip: 15 ip: 3

op: 5*3, 2*1, 1*1 op: 5*3 op: 2*1, 1*1

---

72) Code to accept salary from keyboard and find tax, tax is 5% on salary but **minimum** tax is 200/-

Logic: first calculate tax as 5% on salary, if tax is <200 then replace tax=200 as minimum and print it.

- suppose if salary is 30,000/- then tax is 1500/-, this tax is > minimum , so print 'tax' without changing it
- suppose if salary is 2000/- then tax is 100/-, but we have to collect 200/- as min, so replace tax with 200/-

ip: salary: 30000 ip: salary: 1200 ip:800

op: tax=1500/- op: tax=200/- (minimum) op:200/- (as minimum)

---

73) Accept 3 subject marks and print result pass or fail.

logic: For every student, college is adding 5 marks to one subject which is lowest of 3 subjects. But after adding 5 marks, if it crosses 100 then round off to 100. Finally print pass/fail. For passing, student must obtain>50 in all subjects.

ip: 68 46 77 → 68 56 77 → pass ip: 77 88 99 → 82 88 99 → pass

ip: 98 97 100 → 98 100 100 → pass ip: 47 38 49 → 47 42 49 → fail

---

74) In above programs, we checked the leap-year with simple condition by 4, but for every 400 years one more extra leap year happened. Now find whether given year is leap-year or not?

case1: if year divisible by 400 said to be leap-year (eg:1600,2000 are leap-years but not 1700,1800, 2100)

case2: if year divisible by 4 but not with 100 is also said to be leap-year, for eg:1996,2004,2008

the code is: if(year%4==0 && year%100!=0)

case3: if above two cases are not satisfied then it is not leap year.

ip: 1800 ip: 1600 ip: 1400

op: non leap year op: leap year op: non leap year

---

75) If 3 values are input from KB, write a program to print in ascending order(without sorting a,b,c values)

ip: 12 5 65
op: 5 12 65

```
if(a<b && a<c)                  // if true then → 'a' is 1st small
{       if(b<c)                 // if true then → 'b' is 2nd small
                print(a,b,c);   // printing  a, b, c as ascending order
        else
                print(a,c,b)
}
else  if( b<c)                  // if true then → b is 1st small
{       .....                   // here compare A&C and print B,A,C  or else B,C,A
        .....
}
else
{       .....                   // if control came this block means, C will be the 1st small
        .....                   //  here compare A&B and print C,A,B  or else C,B,A
}
```

---

76) A number N which is b/w 10-100 input through the keyboard and find whether it is prime or not?
case1: If input N is not in limits of 10-100 then show an error message "invalid input". // if(n<10 or n>100)
case2: If N is in limits then find prime-ness by dividing N with 2,3,5,7. If N is divided with any of these numbers then it is said to be **not prime**, or else **prime**

note: Prime numbers divide only with 1 and itself, i.e., it does not divide with any other number such as 2,3,4,5,6,7,8,….N-1. If not divided with these numbers then we can say it is 'prime'.

logic: if N is not divided with 2 then it will not be divided with 4,6, 8,10…(for all 2 multiples)
similarly, if not divided with 3 then it will not be divided with 6,9,12,15…(for all 3 multiples)
So it is better check prime-ness with 2,3,5,7 for N below 100.

ip: 17	ip: 15	ip: 97
op: yes, prime	op: no, not prime	op: prime

---

77) In Japan, population is diminishing, the government encouraging population by cutting down tax to zero those who have more than 1 child. Keep this in mind, write a program to accept employee salary and calculate tax, if salary<=20000 then tax is zero, or else 30% tax on above 20000/- salary.

For example, if salary is 50000 then taxable salary is 50000-20000→30000.
method: For this program, first scan salary as input, if salary<=20000 then set tax to zero, if salary>20000 then scan input for no.of children he has, if children>=2 then set tax=0 or else calculate tax.

For this program input is 'salary' and output is 'tax'

ip: enter salary: 5000	ip: enter salary: 50000	ip: enter salary:50000
op: tax=0	enter no.of children:3	enter no.of children:1
	op: tax=0	op: tax= 9000

---

78) Write a program to find person is eligible for job or not? If age of a person>30 then he is eligible, if age<=30 then scan input for education years, if edu-years>=16 then say 'eligible' otherwise 'not eligible'.

ip: enter age: 49 ip: enter age: 23 ip: enter age: 29
op: eligible enter education years:17 enter education years: 10
op: eligible op: not eligible

---

79) Accept a single number, the number should be 2 or 3 digits, find the number is palindrome or not? if the number and its reverse are equal then it is said to be palindrome. Like 121, 323, 22, etc.

ip: 234 ip: 272 ip: 44
op: not palindrome op: palindrome op: palindrome

---

80) Program to accept a single number(N), the number may have 2 or 3 digits, print its reverse.

ip: 234 ip: 27
op: 432 op: 72

logic: if N<100 then it is 2-digit number or else it is 3-digit number.

---

81) If valid date (d, m, y) is input through keyboard, write a program to decrement it by one day

ip: 1-3-2012
op: 29-2-2012

---

82) find given input date lies in between 4-5-2002 to 7-5-2010 or not?

ip: 1-6-2002 ip: 1-3-2002 ip: 3-4-2010 ip: 8-5-2010
op: yes op: no op: yes op: no

method1:

step1: take lower date as single number → 2002-5-4 → 20020504
step2: take upper date as single number → 2010-5-7 → 20100507
step3: convert input date as single number like: k=y*10000+m*100+d
step4: if(20020504>=k && k<=20100507) printf("it is lying b/w");
else printf("it is not lying b/w");

method2: try without converting into single number.

---

83) If two dates are input from keyboard, write a program to find latest date among them.

Take input dates as 6 values (d1,m1,y1 as date1) and (d2,m2,y2 as date2)

ip: 29-2-2012
30-12-2010
op: date-1 is latest

method2: Compose date1 and date2 (3-values) into single value. (eg: k1=y1*10000+m1*100+d1)
if (d1, m1, y1) are (29, 02, 2012) then k1 would be → 20120229
if (d2, m2, y2) are (30, 12, 2010) then k2 would be → 20101230
Now compare k1 and k2 and find latest (this is simple than method1)

About logical operators

84) When more conditions exist then they must combine using logical operators, otherwise, program may show wrong output or syntax error. Let us observe the output of following program.

```
void main()
{   int k = 5;
    if( 10 < k < 20 )
        printf(" \n  k is in between 10 & 20 ");
    else
        printf(" \n k is not in between 10 & 20 ");
}
```

output: "k is in between 10&20" → wrong output, the evaluation of if-statement is as follows

- if(10 < 5 < 20) // 10 < 5 → false → 0
- if(0 < 20) // 0 < 20 → true → 1
- if(1)
- if(true)

Here the test condition made true hence if-block executed. When two or more relations exist, they must be composed by logical operators. The correct code is:

```
void main()
{   if(10 < k  && k < 20)
        printf("\n k is in between 10 & 20");
    else
        printf("\n  k is not in between 10 & 20");
}
```

The condition is evaluated as if(10<k && k<10) ➔ 10<5 && 5<10 ➔ 0&&1 ➔ 0 (false)

Null instruction caused by semicolon(;)

85) The extra semicolon(;) in the program goes to null-instruction, that is, if anybody by mistake has given extra-semicolon then it forms as null-instruction. For example,

```
a=10;
b=20;
    ;          // observe the extra semicolon
c=30;;         // observe the one more extra semicolon
d=40;
```

here we have 6 instructions not 4, because extra semicolons considered as null-statement by the compiler. This extra semicolon does nothing in this example.

Null instruction as if-statement's body

Observe the following example

86)

```
void main()
{   int A=10, B=20;
    if(A<B)
        ;    // null-instruction
    else
        printf(" sky is blue ");
}
```

Output: no output

87)

```
void main()
{   int A=20, B=10;
    if(A<B)
            ;
    else
        printf(" sky is blue ");
}
```

output: sky is blue

88)

```
void main()
{   int A=10, B=20;
    if(A<B) ;
    else ;
        printf(" sky is blue ");
}
```

output: sky is blue

Guess the output of following program

Remember, if braces are not given for if/else body then compiler considers only one statement.

89)

```
A=10;   B=20;
if( A<B)
     printf("one");
     printf("two");
printf("three");
printf("four");
```

90)

```
A=10;     B=20;
if( A>B)
      printf("one");
      printf("two");
printf("three");
printf("four");
```

91)

```
A=10;     B=20;
if( A<B)
     printf("one");
else
     printf("two");
     printf("three");
printf("four");
printf("five");
```

92)

```
A=10;     B=20;
if( A>B)
     printf("one");
else
      printf("two");
      printf("three");
printf("four");
printf("five");
```

93)

```
A=10;     B=20;
if(A<B)
          ;
else
     printf("two");
     printf("three");
printf("four");
printf("five");
```

94)

```
A=10;     B=20;
if( A<B)
     printf("one");
else
          ;
     printf("two");
     printf("three");
printf("four");
printf("five");
```

95)

```
A=10;   B=20;
if( A<B)
     printf("one");
     printf("two");
else
     printf("three");
```

here we get compile time error called "misplaced else" why?

→

In between 'if' and 'else' keywords, only one instruction is allowed. If more instructions exist then they must be placed in pair of { }. Otherwise, we get error called "misplaced else"

96)

```
void main()
{    int A=10,B=20;
     if(A>B) ;
          printf("one");    → compile time error
     else
          printf("two");
 }
```

→

output: compile time error

In our eye view, it seems to be only one instruction exist in if-body. Unfortunately, it contained two instructions.

First one is: Null instruction happened by semicolon Second one is: printf("one")

we know between if-else keywords there should be only one instruction is allowed.

While loop

1)
```
i=10;
while( i<=20 )
{       printf("%d  " , i );
        i++;
}
```

2)
```
i=1;
while( i<=10 )
{     printf(" hello " );
      i++;
}
printf(" over ");   // prints after completion of loop
```

3)
```
i=1;
while( i<=10 )
{       printf("%d " , i );
        i++;
        i++;
 }
```

4)
```
i=1;
while( i<=10 )
{       i++;
        i++;
        printf("%d " , i );
 }
```

5)
```
i=1;
while(i<=10)
{       printf("%d " , i );
        i=i+2;
}
```

6)
```
i=10;
while(i>0)
{       printf("%d " , i );
        i = i-2;
}
```

7)
```
i=1;
while(i<=5)
{       printf("%d  " , i );
        i++;
}
printf(" last value is  %d " , i );
```

8)
```
i=1
while(i<=5)
{       i++;
}
printf("%d " , i );      // prints after completion of loop
```

9)
```
i=1;
while(i<=10)
{       printf("%d " , i );
}
```

10)
```
i=1;
while(i<=10)
{       printf("%d " , i );
}
i++;
```

11)
```
i=5;
while(i>0)
{       printf("%d  " , i );
        i--;
}
```

12)
```
i=0;
while(i<=5)
{     printf("%d " ,  i + 10  );
      i++;
}
```

13)
```
i=5;
while( i < i +10 )
{       printf("%d " , i );
        i++;
}
```

14)
```
i=1;
while( i<=100000 )
{     printf("%d " , i );
      i=i*10;
}
```

15)
```
i=1;
while(i>10)
{       printf("%d  " , i );
        i++;
}
```

16)
```
i=1;  k=100;
while(i<=10)
{    printf("%d " , k );
     k++;  i++;
}
```

17)
```
n=100;
while(n>0)
{       printf("%d  " , n );
        n=n/2;
}
```

18)
```
p=1;
while(p<100)
{       printf("%d  " , p );
        p=p*2;
}
```

19)
```
i=1;
while( i != 5 )                    // if i not equal 5
{       printf("%d " , i );
        i++;
}
```

20)
```
i=1;
while( i<10 && I != 5 )
{       printf("%d " , i );
        i++;
}
```

21)
```
i=1;
while( i<10 ||  i != 5 )
{   printf("%d " , i );
    i++;
}
```

22)
```
i=5;
while( i%2 == 1 )
{       printf("%d " , i );
        i++;
}
```

23)
```
N=10;
while(N>0)
{       if( N%2 == 1 )
                printf("%d  " ,  N );
        N--;
}
```

24)
```
int N=13;
while( N>1 )
{       printf( "%d  " , N );
        if( N%2==0 )            // if N is even
                N=N/2;
        else
                N=3*N+1;
 }
```

25)
```
i=0;
while( i<5)
{    printf(" hello " );
     if( i<2 )
        printf("world");
     i++;
}
```

26)
```
i=1;
while(i<10)
{       if( i%2 == 1)  x="hello";
        else  x="world";
        printf("%s  " ,  x  );
        i++;
}
```

27)
```
i=1;
while(i<=5)
{    printf("%d  " , i );
     i++;
}

i=5;
while(i>0)
 {        printf("%d  " , i );
          i--;
 }
```

28)
```
i=1;
while( i<=10 )
{     printf("%d " , i );
      if( i==4 )
            i=i+2;
      i++;
}
```

29)
```
i=1;
while( i<=10 )
{     printf("%d " , i );
      if(  i==5 )
            i++;
      i++;
}
```

30)
```
i=1;
while(i<=10)
{       printf("%d " , i );
        if( i==5)
             i=20;
        i++;
}
printf(" %d " , i );
```

31)
```
i=1;
while( i<=10 )
{     printf("%d " , i );
      if(  i==4 )
            i=0;
      i++;
}
```

32)
```
i=1;
while( i<=10 )
{       printf("%d " , i );
        if( i==5 )
              i--;
        i++;
}
```

33)
```
i=1;
while( i<=30 )
{       printf("%d " , i );
        if( i==5 )
              i=i+10;
        i++;
}
```

34)
```
i=0;
while( i<=5 )
{       i=1;
        printf("%d " , i );
        i++;
}
```

35)
```
i=1;
while( i <= 10 )
{       if( i==4  ||  i==6 )
              printf("%d  " , i );
        i++;
}
```

36)
```
i=1;
while(i<=10)
{       if(  ! ( i==4  ||  i==6 ) )  // if  'i'  is not 4 or 6
              printf("%d  " , i );
        i++;
}
```

37)
```
i=1;
while( i<20 )
{       if( i %2 == 1 )
              printf("%d ", i );
        i++;
}
```

38)
```
i=1;
while( i<20 )
{       if( i %3 == 1 )
              printf("%d ", i );
        i++;
}
```

39)
```
i=1;
while( 12%i == 0 )
{       printf("%d ", i );
        i++;
}
```

40)
```
i=1;
while( 12%i == 0 )
{       i++;
}
printf(" output is %d ", i );
```

41)
```
i=1;
while(i<=15)
{       if( 15%i == 0 )
              printf("%d ", i );
        i++;
}
```

42)
```
i=1;
while( 15%i == 0 )
{       printf("%d ", i );
        i++;
}
```

43)
```
i=12;
while( i>0  )
{       if( 12%i ==0 )
              printf("%d ", i );
        i--;
}
```

44)
```
i=17;
while( 18%i != 0 )
{       i--;
}
printf(" output is %d ", i );
```

45)
```
i=2;
while ( 77% i != 0 )
{       printf("%d ",  i );
        i++;
}
printf(" loop stopped at %d " ,  i );
```

46)
```
i=1;  count=0;
while(i<=15)
{       if( 15%i==0 )
              count++;
        i++;
}
printf("%d " ,  count );
```

47)
```
i=2;
while( 35%i != 0 )
{       i++;
}
printf("smallest  factor of 35 is %d" , i );
```

48)
```
N = 35;  i = N/2;
while( N%i != 0 )
{       i--;
}
printf("biggest factor of N is %d" , i );
```

49)
```
i=1;
while( i <=35)
{   if( 35%i==0 )
        X=i;
    i++;
}
printf(" output is %d",  X );
```

50)
```
X=12; Y=18; i=1;
while( i<X )
{     if ( X%i==0 && Y%i==0 )
            printf("%d  " ,  i );
      i++;
}
```

51)
```
i=1;
while(i<20)
{       printf("%d  " , i );
        if( i%3==0 )
              printf("\n");      // goto next-line
        i++;
}
```

52)
```
if(N%2==0)
        N=N-1;
   while( N>0)
   {       printf("%d    " , N);
           N=N-2;
   }
```
ip: if N=13 then op ?
ip: if N=14 then op ?

53)
```
N=77;  i=2;
while( i<=N )
{     if( N%i == 0 )
      {     printf("%d " ,  i );
            i=N;
      }
      i++;
}
```

54)
```
i=2;
while( N%i != 0 )
      i++;
if( i==N)  printf("prime");
else printf("not prime");
```
ip: N=17 then op ?
ip: N=16 then op: ?

55)
```
i=1;
while( i<10 )
    printf("%d  " ,  i++ );
```

56)
```
i=1;
while( i<10 )
    printf("%d  " ,  ++i );
```

57)
```
i=1;
while( ++i < 10 )
    printf("%d  " ,  i  );
```

58)
```
i=1;
while( i++ < 10 )
    printf("%d  " ,  i );
```

59)
```
i=1;
while( i < 10 )
{   printf("%d  " ,  i++  );
    i++;
}
```

60)
```
i=1;
while( i < 10 )
{   ++i;
    printf("%d  " , ++i );
}
```

61)
```
n=7;  i=1;
while(i<=5)
{       p=n*i;
        printf("%d  " , p );
        i++;
}
```

62)
```
i=1;  n=7;
while(i<=5)
{     printf("%d  " , n*i );
      i++;
}
```

63)
```
i=1, n=7;
while( i<=5 )
{     printf("\n %d * %d = %d", n , i , n*i );
      i++;
}
```

64)
```
i=1;  n=7;
while( i<=5 )
{     n=n*i;
      printf("%d  " , n );
      i++;
}
```

65)
```
int N=100 , sum=0;
while( N>0 )
{   sum=sum + N;
    N=N/2;
}
printf("%d " ,  sum);
```

66)
```
i=1; sum=0;
while( i<=4)
{   sum = sum + i*i;
    i++;
}
printf("%d  " , sum);
```

67)
```
i=1;  s=0;
while(i<6)
{      s=s+2;
       i++;
}
printf("%d  " , s );
```

68)
```
i=1;  s=0;
while(i<6)
{      s=s+2;
       printf("%d  " , s );
       i++;
}
```

69)
```
    i=1;  sum=0;
while( i<=4 )
{     sum=sum+i;
      printf("%d   " , sum );
      i++;
}
printf("%d   " , sum );
```

70)
```
i=1;  s=7;
while( i<6 )
{      printf("%d  " , s );
       s = s * -1;
       i++;
}
```

71)
```
i=1;  s=1;
while(i<6)
{   printf("%d  " , s );
    s=s*2;
    i++;
}
```

72)
```
i=1;  x=2;
while(i<5)
{   printf("%d  " ,  x );
    x=x*x;
    i++;
}
```

73)
```
p=i=1;  x=2; y=3;
while( i<=y )
{   p=p*x;
    i++;
}
printf("%d  " , p );
```

74)
```
i=1;  f=1;
while(i<6)
{   printf("\n fact of  %d  is  %d  " , i , f );
    i++;
    f = f*i ;
}
```

75)
```
i=3; s=0;
while(i<7)
{   s=s*10+i;
    i++;
}
printf("%d  " ,  s );
```

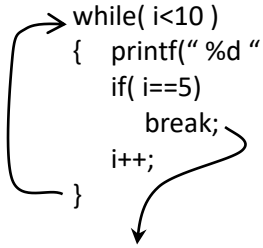
76)
```
int i=3; s=0; p=1;
while(i<7)
{   s=s+p*i;
    p=p*10;
    i++;
}
printf("%d  " , s );
```

77)
```
i=1;  sum=0; p=1;
while(i<=5)
{   sum=sum+p;
    p=p*2;
    i++;
}
printf("%d  " , sum );
```

78)
```
int sum=0 ,  N=2345;
while(N>0)
{   sum=sum+N%10;
    N=N/10;
}
printf("\n  sum  is   %d" , sum );
```

79)
the **"break"** is a keyword, stops the loop in the middle of execution, let us see following example

```
i=1;
while( i<10 )
{   printf(" %d " , i );
    if( i==5)
        break;        // break throws the control out
    i++;
}
```



Output: 1 2 3 4 5

80)
The relational operator gives Boolean value (true/false)
1<2 → true → 1
2<1 → false → 0

```
while( 1<2 )  //  always true, infinite loop
{       printf(" hello ");
}

while( 1>2 )  // always false, no looping, no-output
{       printf(" world ");
}
```

In c, in condition place, the value zero is taken as false, and other than zero is taken as true.
so the value 0 is false, whereas 1, 2, 0.5, -3, 10, -1.4 , 0.23 are considered to be true

```
while(0)      // zero means   false
{   printf("wolrd");
}

while( 1 )    // always true, infinite loop
{   printf("world");
}
```

```
while( -1.2)   // always true
{   printf(" comes here again and again");
}

while( -0.2)   // always true
{   printf(" comes here again and again");
}
```

81)
```
N=5;
while(N)                  // stops when N gets 0
{   printf(" %d " , N );
    N--;
}
```
op: ?

82)
```
N=1;
while( N!=5 )
{   printf("%d  " , N );
    N++;
}
```
op: ?

83)
```
i=1; count=0;
while(1)
{     if( 35%i == 0 )
          count++;
      if( count==2)
          break;
      i++;
}
printf("second factor of 35 is %d" , i );
```

84
```
int N=34567 , count=0;
while( N>0 )
{   R=N%10;
    printf("%d ", R );
    N=N/10;
}
```

85)
```
int N=34567;
while( N>0 )
{   R = N%10;
    if( R%2 ==0 )
        printf( "%d  " , R );
    N=N/10;
}
```

86)
```
int N=3456;
while( N>9 )
{   N=N/10;
}
printf( "%d " , N );
```

87)
```
int  N=34567 , count=0;
sum=N%10;
while( N>9 )
{    N=N/10;
}
sum=sum+N;
printf( "sum is %d " , sum );
```

88)
```
int N=345 ,  prod=1;
while( N>0 )
{     R = N%10;
      prod=prod*R;
      N=N/10;
}
printf(" %d  " , prod );
```

89)
```
int N=34567,  sumOdd=0 , sumEven=0;
while( N>0 )
{    R = N%10;
     if( R%2 == 1 )  sumOdd=sumOdd+R;
     else   sumEven=sumEven+R;
     N=N/10;
}
pf("sums are %d   %d" , sumOdd, sumEven);
```

90)
```
 int N=345, rev=0;
while(N>0)
{     R=N%10;
      rev=rev*10+R;
      N=N/10;
}
printf(" %d  " , rev );
```

91)
```
int N=34567  ,  new=0 ,  p=1;
while( N>0 )
{    R = N%10
     if( R%2 ==1 )
     {      new=new+p*R;
            p=p*10;
     }
     N=N/10;
}
printf(" new number is %d" , new );
```

92)
```
int N=3745, big=0;
while(N>0)
 {     R=N%10;
       if( big<R )
             big=R;
        N=N/10;
 }
 printf(" %d  " ,  big );
```

93)
```
i=0;
while(i++ < 10 )
     printf(" hello ");
     printf(" world ");
```

94)
```
for( i=0;  i<10;  i++ )
 {    printf("hello");
      printf("world");
 }
```

95)
```
i=5;
for( i++;  i<11 ;  i++ )
{     printf("hello");
}
```

96)
```
i=1;
for(   ; i<11 ;    )
{    printf("hello");
     i++;
}
```

97)
```
for(   ;    ;    )        // infinite loop
{    printf(" hello ");
}
```

98)
```
for(  i=1 ;      ; i++ )      // this is also a infinite loop
{    printf("%d  " , i  );
}
```

99)
```
for(  i=1;   1   ; i++   )  // '1' represents always true
{    printf(" %d " , i );
}
```

100)
```
 for( i=1;   i<10    ; i++    )
 {    printf(" %d  " , i );
      i++;
 }
```

101)
```
 for( i=10;   i<=100;   i=i+2 )
      printf(" %d  " , i );
```

102)
```
for( i=10;   i  ;  i-- )  // stops when 'i' is 0, since 0 is false.
     printf(" %d  " , i );
```

103)
```
 for( i=1, j=10 ;   i <=10 ;   i++, j++ )
      printf(" %d  " ,  j );
```

104)
```
 for( i=1, j=10 ;   i<j;   i++, j--)
      printf("\n %d  %d " ,  i , j );
```

Loops

write first 20 programs using while-loop and for-loop, remaining programs is your choice.
Actually both loops performance is same, so we can use any loop as per your convenient.

---

1) Code to print numbers from 10 to 20, here no scanf() is required, because output is fixed limits.
op: 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20

---

2) Code to print "Hello" for 10 times. (do not take any input from KB)
op: Hello Hello Hello Hello … 10 times

---

3) scan two numbers as lower & upper limits and print numbers between them
hint: take 'L' & 'U' as input variables and print numbers from L to U.
ip: 10 , 20
op: 10 11 12 13 14 15 16 17 18 19 20

ip: -12 , 3
op: -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3

---

4) scan two numbers as lower & upper limits (L & U) and print numbers from upper to lower in reverse order
ip: 10 , 20
op: 20 19 18 17 ….. 10

---

5) scan two numbers as lower & upper limits (L & U) and print all 3 divisible numbers between them
ip: 10 , 20
op: 12 15 18

ip: 9 25
op: 9 12 15 18 21 24

---

6) Code to print N-5 to N+5 , here **'N'** is input value, take from keyboard. Here lower limit is **N-5**, upper is **N+5**
ip: if N is 50
op: 45 46 47 48 49 50 51 52 53 54 55

---

7) Code to accept **'N'** from keyboard and print 10 numbers before & after of a given number.
ip: enter N value: 45
op: 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55

---

8) Code to print N to 1, here N is input value, take from keyboard. Here upper limit is N, lower is limit is 1
ip: if N is 8
op: 8 7 6 5 4 3 2 1

---

9) Code to accept N from keyboard and print odd numbers from 1 to N.
ip: enter n value: 15
op: 1 3 5 7 9 11 13 15
method: take loop variable 'i' with 1, and increment it by 2 to get next odd value in the loop.

---

10) print all 3 divisible numbers from N to 1, here N is input value
ip: 20
op: 18 15 12 … 1

---

11) Code to print odds from N to 1

ip: if N is 20

op: 19 17 15 13 11 … 5 3 1

step1: scan N as input

step2: if N is even then do N-- // because we have to start with odd number. If N is divisible by 2 then it is even

step3: now take while loop and print odds as given below

```
while(N>0)
{       print N as odd output
        decrement N by 2  to get next odd number
}
```

step4: stop

---

12) Code to print 1, 2, 3, 4, 5, ...10 and also print 10, 9, 8, 7, …,3, 2, 1.

Here no input statement is required, i.e, scanf() is not required as we need to print fixed no.of 10 times.

Here the two series values should be separated with the word **'and'**

logic: Write two loops, loop after loop, the first loop prints 1-to-10, whereas second loop prints 10-to-1.

op: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ***and*** 10, 9, 8, 7, 6, 5, 4, 3, 2, 1

Note: add this printf("and") statement between two loops

---

13) Accept N from KB, if N is even then print "Hello" for 10 times, if N is odd then print "World" for 10 times.

ip: N is 12 (even) ip: N is 13 (odd)

op: Hello, Hello, Hello … 10 times op: World, World, World … 10 times

---

14) Code to print numbers 1, 10, 100, 1000, 10000, 100000. (6 numbers)

---

15) Code to print numbers 100000, 10000, 1000, 100, 10, 1. (6 numbers)

---

16) Code to print N, N/2, N/4, N/8,….1, here N is input from keyboard

ip: N=100

op: 100, 50, 25, 12, 6, 3, 1

---

17) Code to print 1, 2, 4, 8, 16, 32, 64, ……<N, where N is input from keyboard

ip: N=300

op: 1 2 4 8 16 32 64 128 256

---

18) Program to print 7, -7, 7,-7, 7, -7, …N times (here N is input value)

logic: take 'i' for the looping N times, increment it by 1 for N times.

take 'V' with 7 and print in the loop, multiply 'V' with -1 to change its sign for next cycle.

here, the sign of 'V' alternatively changes to +ve to -ve and –ve to +ve.

---

19) Code to print 5#9, 7#14, 9#19, 11#24, 13#29, …. for 10 times.

logic: write printf() statement as printf("\n%d # %d ", x , y); where 'x' increments by 2 and 'y' by 5.

take 'i' for looping 10 times, starting values of x is 5, y is 9

---

20) Code to print all multiples of N, which is as given below.

ip: if N is 7 ip: if N is 9

op: 7, 14, 21, 28, 35, 42, …. up to 10 times op: 9, 18, 27, 36, …. up to 10 times

---

21) Code to find sum of 2+2+2+2+2+ ….N times. Here N is input value.
Condition: do not use multiplication operator (*) in the program.
ip: enter N value:5
op: output = 10
hint: sum=sum+2 // repeat this instruction for N times to get final output value.

---

22) Code to find sum of X+X+X+X+X+ …. N times. Here X , N are input values.
Condition: do not use multiplication operator (*) in the program.
ip: enter X , N values: 3 5
op: output = 15

---

23) The sum of squares of the first ten natural numbers is $1^2 + 2^2 + 3^2 + 4^2 \ldots + 10^2 = 385$
Write a program to prove it (output is "right answer" or "wrong answer")
The logic like: sum=sum+i*i, where i=1 to 10 do
if sum==385 then print "right answer" else print "wrong answer". Do not take any input from KB.

---

24) Code to find sum of 1+2+4+8+16+32+64 ... N times.
note: do not use pow() library function.
ip: enter N value: 5
op: sum=31
The logic like: sum=sum+p; here p = 1, 2, 4, 8, 16, 32, 64, ….

---

25) program to find sum of N + N/2 + N/4 + N/8 + … + 1. N is input value
ip: 10
op: 18 (10+5+2+1)

---

26) Program to find sum of 1+2+3+4+5+ ……+N. Here N is input value
note: do not use formula like N*(N+1)/2
ip: N = 5
op: sum=15

---

27) Program to find sum of 1/1 + 1/2 + 1/3 + 1/4 + 1/5 + …… + 1/N. Here N is input value
ip: N = 5
op: sum=2.283 (1.09 + 0.50 + 0.33 + 0.25 + 0.20 → 2.283)

---

28) Program to find sum of 1/2 + 3/4 + 5/6 + 7/8 + 9/10 + …… + (2*N-1)/(2*N). Here N is input value
ip: N = 5
op: sum=3.858 (0.500 + 0.750 + 0.833 + 0.875 + 0.900 → 3.858)

---

29) Program to find sum of (1) + (1+2) + (1+2+3) + (1+2+3+4) + …. N times. Here N is input value
ip: N = 5
op: sum = 35 (1 + 3 + 6 + 10 + 15 → 35)
logic: sum1=sum1 + i ; // here 'sum1' values in each iteration will be : 1, 1+2, 1+2+3, 1+2+3+4, …
sum2=sum2+sum1 ; // add this 'sum1' values 'sum2' to get final value. Here " i " value are 1,2,3,4,5,…

---

30) Code to accept numbers one by one until zero as end, when zero is entered then stop scanning and print sum of all values.

Ip: 12↵
10↵
4↵
2↵
0↵ (stop)
op: sum=28 (12+10+4+2)

Here, the scanf() statement needs to be written inside the while loop because we need to scan continuously until the last input is zero. Use an infinite loop as while(1) { ... } and include a break statement to stop the loop when the input is zero. The value 1 used as the loop condition represents an infinite loop — of course, the loop stops when break is encountered.

```
void main()
{       int N, sum=0;
        while(1<2)  or while(1)          // this loop is said to be always true( infinite loop), but stops by 'break'
        {       printf("enter N value :");
                scanf("%d", &N);
                if(N==0)
                    break;               // here this 'break' throws the control out of loop, ie, it stops the loop
                here add each N value to 'sum'  // because we need to find sum of all inputs
        }
        print final sum value 'sum'.
}
```

---

31) Code to find product of 2*2*2*2*2….N times. Here N is input value. Here do not use pow() function

ip: enter N value:5
op: output = 32

---

32) Write a program to find X^5. Here X is input value. Multiply X*X*X*X*X for 5 times. Do not use pow().

ip: if X is 3
op: 243 (3*3*3*3*3)

---

33) If base(x) and exponent(y) are input through the keyboard, write a program to find x^y.

note: do not use pow() function.

ip: enter x , y values: 2 3
op: output of 2^3=8

logic: multiply x*x*x … Y times

---

34) Write a program to find factorial of N. Here N is input value.

ip: if N is 5
op: 120 (1*2*3*4*5)

---

35) Program to find sum & product of 1 to N [1+2+3+....+N , 1*2*3*....*N]

ip: if N is 5
op: sum=15
product=120 (using single loop we can do it)

---

36) Code to print 1, 2, 4, 8, 16, 32, … N terms. The N is input taken from keyboard.
These values are nothing but power 2 series: 2^0, 2^1, 2^2, 2^3, 2^4, 2^5, …N times.

logic: take the variable 'i' for looping N times
take the variable 'p' to produce and print 1, 2, 4, 8, 16, 32,…

---

37) Code to find sum of 1+2+4+8+16...N times. ($2^0+2^1+2^2+2^3+$ …. N times)

note: do not use pow() library function.

ip: enter N value: 5
op: sum=31

---

38) print values of X^1 , X^2 , X^3 ..... 5 times [do not use pow() fn]

ip: x=2
op: 2, 4, 8, 16, 32

ip: x=3
op: 3 , 9, 27, 81, 243

---

39) find sum of values of $X^1 + X^2 + X^3$ ..... 5 times [do not use pow() fn]

ip: x=2
op: 62 (2+4+8+16 + 32)

ip: x=3
op: 393 (3+9+27+81+ 273)

---

40) $X^1/1! + X^2/2! + X^3/3!$..... 5 times [do not use pow() fn , the symbol '!' do not works like factorial

ip: x=3
op: sum= 17.4 [3.0 + 4.5 + 4.5 + 3.375 + 2.025 → 17.4]

logic:
step1: take p , fact , x, sum, i as variables.
step2: scan input values to 'x' .
step3: initialize starting values for sum=0, p=1, fact=1
step4: take loop and repeat this step for 5 times (for i=1 to 5 do)
multiply 'p' with 'x' and also multiply 'fact' with 'i'.
add p/fact to sum.
repeat loop
step5: print(sum)
step6: stop

---

41) If the number **N** is input through the keyboard, write a program to print all factors of **N** and also count total number of factors.

logic: to find factors of N, check N by dividing with all possibility from 1, 2, 3, 4 ...N.
if N%i==0 then 'i' is a factor of N. Let N=15 then loop repeats as given below
if(15%1==0) is true, so 1 is factor of 15
if(15%2==0) is false, so 2 is not factor of 15
if(15%3==0) is true , so 3 is factor of 15
if(15%4==0) is false , so 4 is factor of 15
in this way check with all possibilities from 1 to N.

ip: enter **N** value:15
op: 1, 3, 5, 15
Count of factors=4

---

42) scan two numbers as lower & upper limits and print 3 and 5 divisible between them

ip: 10 , 20

op: 10 (5 divisible)
12 (3 divisible)
15 (3,5 divisible)
18 (3 divisible)
20 (5 divisible)

---

43) If N is input through the keyboard, write a program to print small factor other than 1.

ip: enter N value:18 ip: enter N value:15 ip: enter N value:35

op: output is:2 op: output is:3 op: output is:5

logic: for small factor, divide N with 2,3,4,5,…N, that is, check with all possibilities from 2 to N, which ever divides first then it is small factor and stop the loop.

---

44) Write a program to print big factor **other than N**. Here **N** is input value.

Hint: Generally, for any number, the possible factors lie between 1, 2, 3, … N/2.

That is, there should not be any factors after N/2 except N itself. For example, if we take 18, then the factors are 1, 2, 3, 6, and 9. The last factor is 9, so there should not be any factors found after 9 (except 18 itself).

So, the value 18 is never divisible by 10, 11, 12, 13, … 17. So it is useless to check for factors after N/2.

Our requirement is to find the biggest factor other than N, so it is wise to check for factors from N/2 down to 1.

Logic: Divide N by numbers from N/2 down to 1. That is, check all possibilities from big to small; whichever divides first is the biggest factor. (So, check from i = N/2 to 1.)

---

45) Program to accept two numbers from keyboard and print their common factors

ip: 12 18 ip: 10 30

op: 1, 2, 3, 6 op: 1, 2, 5, 10

step1: take two input values into x , y

step2: find smallest of x , y // because common factors exist from 1 to smallest of x,y

```
if(x<y) small=x
else small=y;
```

step3: repeat while from i=1 to small

```
if(x%i==0 && y%i==0)
    print 'i' as common factor
```

---

46) Program to accept two numbers from keyboard and print biggest common factor (GCD)

ip: 12 18 ip: 10 30

op: 6 op: 10

step1: take two values into x,y

step2: find smallest of x,y // because biggest common factor exist from small to 1

step3: repeat loop from 'small' to 1 (for i=small to 1)

```
if(x%i==0 && y%i==0)
    print( 'i' as common factor )
    break;                    // whichever divides for first time then it will be GCD, so stop the loop
```

step4: stop

---

47) Program to accept a number N and find whether it is perfect or not.

Perfect number: If the sum of all factors (excluding N itself) is equal to the given number N, then it is said to be a perfect number. Logic is, check for factors from 1 to N/2 and add all the divisible numbers to the variable sum.

For example: 6 (1+2+3→6), 28(1+2+4+7+14→28)

ip: enter N value: 6 ip: enter N value: 7 ip: enter N value: 28
op: yes op: no op: yes

---

48) In examinations, interviews, viva, etc., there is always one program that is frequently asked — none other than the "prime number logic". Write a program to find whether the given number N is prime or not.

Primes do not have factors except 1 and itself (they are not divisible by any number other than 1 and itself).

ip: n = 17 ip: n = 18
op: yes, it is prime op: no, it is not prime

logic1: As we discussed in previous problems, for any kind of number, possible factors lie between 2 to N/2. So it is wise to check for prime-ness of N by dividing from 2 to N/2. During checking process, if N is divided then stop the loop and say "not prime". If it is not divisible at all till the end of the loop, then say "prime."

A beginner often writes the prime number logic in the <u>wrong way</u> as

```
for( i=2; i<=N/2; i++)
{       if(N%i==0)
            printf(" not prime");
        else
            printf(" prime");
        i++;
}
```

Here at every division in the loop, we are printing prime/not-prime, so we get many outputs as given below

```
ip:  N=15          ( many outputs )
op:  prime         // when 15%2==0  is false
     not prime     // when 15%3==0  is true
     prime         // when 15%4==0  is false
     not prime     // when 15%5==0  is true
     prime         // when 15%6==0  is false
     prime         // when 15%7==0  is false
```

note: to solve this problem properly, better to use boolean logic, there several other logics to find prime-ness, but this Boolean logic is standard and best, the code as given below

```
bool=1;                 // let us assume N is prime, so take bool as 1 (true).
for(i=2;  i<=N/2;  i++)
{       if(N%i==0)
        {       bool=0;     // here N divided, therefore N is not prime, so set bool to 0 and stop the loop
                break;
        }
}
if(bool==1) printf("prime");
else printf("not prime");
```

logic2: Count all factors of N, i.e., divide N with all numbers from 1 to N and count all factors. If factors count==2 then say it is "prime" or else "not prime". (This is simple logic but takes much time for executing)

---

49) Write a program to find sum of all digits in a given number.

ip: 2345 ip: 456 ip: 23456

op: 14 (2+3+4+5) op: 15 (4+5+6) ip: 20 (2+3+4+5+6)

Logic: Extract digits one by one from N and add them to sum. To do this, divide N repeatedly by 10 and collect the remainders — the remainders are nothing but the digits of N from last to first. Add these digits to 'sum'.

Process is as follows

step1: divide N with 10 and get the remainder(Reminder is always the last digit of N when we divide with 10)

R=N%10 (gives last digit of N as remainder)

step2: add this remainder 'R' to 'sum'

sum=sum+R (adding to sum)

step3: remove the current last digit from N so that we can get the next digit of N in the next cycle

N=N/10 (removes last digit from N)

step4. Repeat these three steps as long as N>0.

---

50) Write a program to print only 'odd' digits in a given number. (print right to left in reverse order)

ip: 2345 ip: 3546 ip: 23457

op: 5 3 op: 5 3 ip: 7, 5, 3

Process is as follows

step1: divide N with 10 and get the remainder(Reminder is always last digit of N when we divide with 10)

R=N%10 (gives last digit)

step2: if R is odd then print on the screen

print (R)

step3: remove current last digit from N, so that we can get next digit for next cycle

N=N/10 (removes last digit from N)

step4. Repeat these 3 steps as long as N>0

---

51) Write a program to print only 'odd' digits in a given number. (print left to right)

Let assume the input number N contained exactly 4-digits.

ip: 2345 ip: 3467 ip: 2345

op: 3 5 op: 3 7 ip: 3, 5

Process is as follows

step1: repeatedly divide the N with 1000,100,10,1 to get digits one by one from left to right.

R=N/D; // here D=1000, 100, 10, 1

step2: if R is odd then print on the screen

print (R)

step3: now remove current first digit from N, so that we can get next digit for next cycle

N=N%D (removes first digit from N)

step4: decrement D to D/10

step5. Repeat these 3 steps as long as N>0

---

52) Program to find the digit '5' exist in a given number or not?

ip: 4356	ip: 346	ip: 4455
op: 5 is exist	op: 5 is not exist	op: 5 is exist

logic: Extract digit by digit from N and check whether it is '5' or not, finally print the result.

A beginner write logic this logic as bad as (Wrong)

```
while(N>0)
{       rem=N%10;     // get last digit in N.
        if(rem==5)      // check it is 5 or not
                printf("5 is exist ");
        else   printf("5 is not exist");
        N=N/10;         // remove current last digit from N, this is to get next digit in next cycle
}
```

If input N is 2356, then above program shows wrong output as (actually we want only single output)

output is: when rem==6 then shows '5 is not exist'
when rem==5 then shows '5 is exist'
when rem==3 then shows '5 is not exist'
when rem==2 then shows '5 is not exist'

solution: use 'boolean' or 'count' logic to solve this problem, this is as said in above programs.

---

53) Write a program to find sum of even & odd digits separately in a given number.

ip: 12453
op: even digits sum = 6 (2+4) , odd digits sum = 9 (1+5+3)

logic: Take two variable to sum up separately, for example, sumEvens, sumOdds

---

54) If a number N is input through the KB, write a program to find whether it is an Armstrong number or not.

Logic: If the sum of the cubes of all digits of N is equal to N itself, then it is called an Armstrong number.

Example: 153 → (1×1×1) + (5×5×5) + (3×3×3) = 153

input: 153	**input:** 445
output: "yes, Armstrong"	**output:** "Not an Armstrong"

Logic: After the loop completes, the value of N becomes 0 because in the loop, the instruction N = N / 10 reduces N to zero. So, before the loop, store the value of N in a variable like 'temp'. After calculating the sum of the cubes in the loop, compare 'temp' with sum to check whether the number is an Armstrong number.

```
scan N;
temp=N;         // saving N to temp before loop
while( N>0 )
{   R=N%10;
    sum = sum + R*R*R;
    N=N/10
}
------
```

---

55) Write a program to find big & small digit of a given number.

ip: 2715

op: big=7 , small=1

logic: take a variable called 'big' with value zero, now compare each digit(R) with 'big' ,
if big<R then take R into 'big', finally 'big' contains biggest value. Likewise find 'small' also.
Remember, the starting value of 'big' is zero and 'small' is 9.

---

56) Write a program to find reverse of given number

ip: 2345

op: 5432

step1: Let N be the input number. Take `REV` to store the reversed value. Initially, set `REV` to zero.

step2: Get the last digit(R) from N and insert it into REV by doing REV=REV*10+R

step3: To get the next digit from N, now remove the current last digit of N by doing N=N/10

step4: repeat step2, step3 until N>0

REV = REV * 10 + n%10

= 0 * 10 + 5 → 5

= 5 * 10 + 4 → 54

= 54 * 10 + 3 → 543

= 543 * 10 + 2 → 5432

= 5432

---

57) Write a program to find whether the given number is palindrome or not.
If the number and its reverse are equal then it is said to be palindrome.

ip: 1221 ip: 12345

op: palin-drome. op: not palin-drome

After the loop, the value of N becomes 0 because the instruction N = N / 10 reduces N to zero during the loop. So, before the loop, store the value of N in a variable like temp. After finding the reverse value in the loop, compare temp with the reverse value to check whether the number is a palindrome.

---

58) Write a program to find given number is valid binary or not?

ip: 1101 ip: 1201 ip: 14011

op: yes, valid binary op: no, not valid binary op: no, not valid binary

logic: extract digit by digit from N, if any digit>1 then stop the loop and say "it is not valid binary".
or else say 'it is valid binary'.

---

59) Write a program to find decimal number from a given binary number

ip: 1101

op: 13

1	1	0	1	→	$1*2^3$	+	$1*2^2$	+	$0*2^1$	+	$1*2^0$	→ 13
2^3	2^2	2^1	2^0		8	+	4	+	0	+	1	

step1: multiply all digits of N with 2^0, 2^1 , 2^2 , 2^3 , 2^4… from right-to-left and sum all these values.

step2: The sum of all these products forms a decimal number.

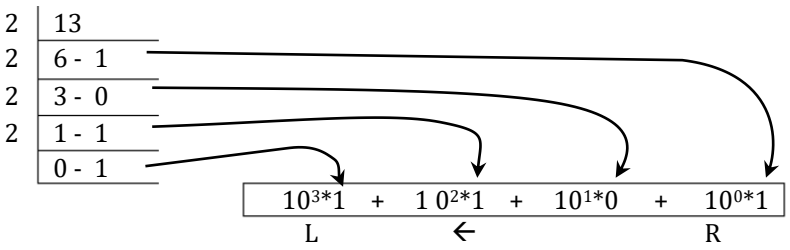
note: To get the values of 2^0 , 2^1 , 2^2 , 2^3…, take 'p' and multiply it with 2 . (do not use pow() function)

---

60) Write a program to find binary number of a given decimal number

ip: 13

op: 1101



logic: divide continuously N with 2, and collect the remainders(R), after getting each R, multiply with $\mathbf{10^i}$ (P) and add to 'sum' variable as shown pic. Here P=1,10,100,1000, …. Do not use pow() fn

---

61) Write a program to print Fibonacci series up to 10 terms.

process: The first two terms in the series are 0, 1 and remaining values are generated by adding previous two values: 0 1 1 2 3 5 8.....

step1: Let us take first two terms as x=0, y=1;

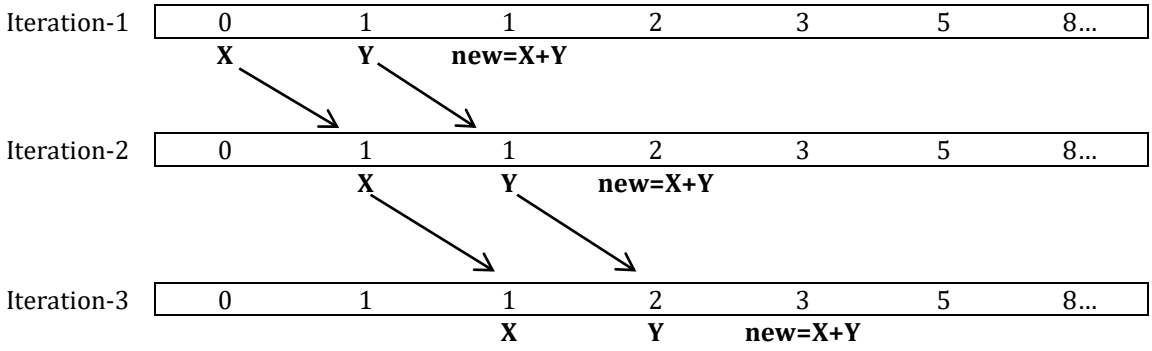
step2: print 'x' as series value(s)

step3: Generate next term by adding **'x+y'** to '**new'**

step4: Now shift '**x'** to '**y'** and '**y'** to **'new'** for next cycle, this is as given below picture.

step5: Repeat step2, step3, step4 for 10 times

Let us see how the x, y are advancing in every iteration of loop

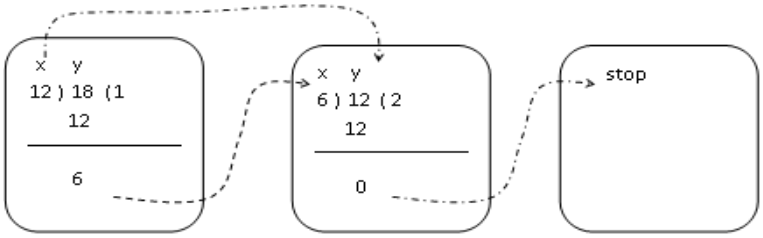


---

62) Write a program to find GCD of two numbers. (GCD/HCF → Greatest Common Divisor)

input: 12, 18 then output is: 6

1. let x, y are input values
2. divide 'y' with 'x' (irrespective of which is big and which is small)
3. if remainder(R) is zero then stop and print 'x' as GCD
4. if R is not zero then take 'x' as 'y' and 'R' as 'x' and continue this process until R is zero.




```
logic:  while(1)     // always true
        {     R=Y%X;
              if( R==0 )
                 break;
              -------
        }
        -------
```

Home Work

70) Code to print multiplication table N, the table should be displayed in the following format.

ip: enter table number: 9

op: 9*1=9
9*2=18
....
9*10=90

hint: write printf() statement as printf("\n %d * %d = %d", n, i, n*i);

---

71) Code to accept **N** from keyboard and print numbers from 1 to N.

Here the last value (N) should be prefixed with the word **'and'.**

ip: enter N value: 14

op: 1 2 3 4 5 …….13 and 14.

---

72) Code to print 1 to 10 values by skipping 5'th value (No input is required).

op: 1 2 3 *4 6* 7 8 9 10.

---

73) Code to print 1 to 100 numbers by skipping from 50 to 60. (No input is required).

op: 1 2 3….46 47 48 49 61 62 63 ….99 100

---

74) Code to print 1 to 20 numbers by skipping 5, 6 and 7 numbers.

For this program, the scanf() is not required, because the target number N is 20, it is fixed value.

op: 1 2 3 4 8 9 10 11 12 13 14 15 16 17 18 19 20.

logic: it is better to use if-statement inside while-loop to skip these values.

for example: If(! (i==5 || i==6 || i==7)) then print(i) // the symbol "!" is called not operator

or if(i==5) then i=i+3; // it means : if ' i ' is not of (5,6,7) then print(i)

---

75) If two input values taken from keyboard as lower and upper limits, write a program to print numbers from lower to upper. For example,

ip: enter lower & upper limits: 14 23

op: 14 15 16 17 18 19 20 21 22 23

Sometimes the input values may entered in reverse order, for example 23, 14 (here lower>upper)

In this case, swap lower & upper before printing. (before loop)

ip: enter lower & upper limits: 30 14

op: 14 15 16 17 18 19 20 21 22 23 25 27 29

step1: Take variable names L,U

step2: scan L,U

step3: if(L>U) then swap using temp;

step4: now print numbers from L to U using while loop

---

76) If two input values is taken from keyboard as lower and upper limits, write a program to print odd numbers from lower to upper. For example,

ip: enter lower & upper limits: 14 23

op: 15 17 19 21 23

Sometimes the input values may enter in reverse order, for example 23, 14 (here lower>upper)

In this case, swap lower & upper before printing. (before loop)

step1: Take variable names L , U

step2: scan L,U

step3: if(L>U) then swap using temp;

step4: if(L is even) then L++; // because we need to start with odd number

step5: now print odd numbers from L to U using loop, here increment L every time by 2 to get next odd.

---

77) Code to accept two limits as lower and upper(L,U) from keyboard and print numbers between them.

If user entered L<U then print in ascending order, but if user entered L>U then print in descending order.

ip: 15 32

op: 15 16 17 18 19 21 22 23 24 25 26 27 28 29 30 31 32

ip: 32 15

op: 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15

- if L<U then print numbers from L to U using increment loop.
- if L>U then print numbers from L to U using decrement loop.

Take if-else statement, in if-block write increment loop, in else-block write decrement loop.

Following example
```
if(L<U)
    while(L<U) { ... }
else
    while(L>U) { ... }
```

---

78) If N is input through the keyboard, write a program to print following output.

ip: N=18

op:
```
1   2   3   4   5
6   7   8   9   10
11  12  13  14  15
16  17  18
```

logic: print new line('\n') for every 5 values, for example, if(i%5==0) printf("\n");

---

79) scan 2 values from KB and print total of them. There is one condition while inputting, the two input values should not be same. If second entered value is also same then show "error, same value entered" and scan again until different value is entered.

ip: 12↵
12↵ error, same value entered, try again
13↵ (stop)
op: 12+13→ 25

ip: 10↵
10↵ error, same value entered, try again
10↵ error, same value entered, try again
15↵ (stop)
op: 10+15→ 25

---

80) In a school, every cricket player student has to face 4 balls, and if he blasted at least one six shot out of 4 balls and also total score is above 10 then he get 10,000/- as the prize money or else zero.

(if input is -1 then player is out in the middle, so stop scanning runs)

enter runs made at ball1: 3	enter runs made at ball1: 2
enter runs made at ball2: 0	enter runs made at ball2: 0
enter runs made at ball3: 6 (six shot)	enter runs made at ball3: 4
enter runs made at ball4: 6 (six shot)	enter runs made at ball4: 2
op: played 6 shot+score>=10, so prize is 10,000/-	op: no prize money

---

81) Code to accept hours**(H)** in time from keyboard and **H** must be in between 0 to 23. If user entered mistakenly H<0 or H>23 then show an error message called "wrong input" and scan H value again and again until he entered a valid hours. (It is much like scanning wrong password again & again).

Later print: good Morning(if H→0 to 12), good Afternoon(H→ 13 to 16) , good Evening(H→17 to 23)

ip: enter H in time :50
op: wrong input, try again

ip: enter H in time :-2
op: wrong input, try again

ip: enter H in time: 8 (input is right)
op: Good Morning

```
logic:   while(1)
         {     printf("enter H value of time 0 to 23 :");
               scanf("%d", &H);
               if(H<0 || H>23)
                     print("invalid input , try again\n");
               else   input is valid, so stop the loop
         }
         // after scanning proper input in the above loop now print following output
         if( H<12)
            printf("good Morning");
         -----
```

---

82) Code to accept a value N from keyboard and N must be between 1 to 20; if user entered mistakenly N>20 then show an error message called "wrong input" and scan N value again and again until he entered a valid number. If N is valid then print multiplication table for N.

ip: enter N value: 50
op: wrong input, it must be in 1 to 20, try again
ip: enter N value: 22
op: wrong input, it must be in 1 to 20 , try again
ip: enter N value: 8 (input is right)
op: 8*1=8
8*2=16
8*3=24 …

method: Write 2 loops, loop after loop (not nested loop), the first loop to scan proper value (for 1-20) and second loop to print 'multiplication' table for 10 terms.

---

83) Program to print 7, -7, 7,-7, 7, -7, …N times

---

84) Program to add all these values (7) + (-7) + (7) + (-7) + …for 10 times;
if the output sum value is zero then display "program is good"
if not then display "program has some logical mistake"

---

85) Program to print 1, -2, 3, -4, 5, -6, 7, …N times

---

86) Code to print 2^0, 2^1, 2^2, 2^3, 2^4, 2^5, …N times. (1, 2, 4, 8, 16, 32 ……for N times)
ip: N=5
op: 1 2 4 8 16 (N terms, here N is 5)

---

87) Code to print X^0, X^1, X^2, X^3, X^4, X^5,… N times.
Here X, N are input numbers, If X is 2 then the output is like above program.
logic: take loop variable 'i' to repeat N times (Increment 'i' every time by 1)
take variable 'p' to produce X^0, X^1, X^2, X^3,X^4, … (multiply p with x to get next value, like p=p*x)

---

88) Code to find sum of $X^0 + X^1 + X^2 + X^3$ + ….. 5 times. Hint: do not use pow() function.
ip: enter X value : 2
op: sum=31

---

89) Program to print sum of each term value 1, 1+2, 1+2+3, 1+2+3+4, 1+2+3+4+5, … N times.
ip: enter N value : 7
op: 1, 3, 6, 10, 15, 21,28
note: do not use nested-loop (using single loop we can solve it)
hint: print 'sum' value inside loop, where sum=sum+'i'; // i=1,2,3,4, ….N

---

90) Find sum of (1) + (1+2) + (1+2+3) + (1+2+3+4) + ...N times (do not use any formula or nested loop)
(1) + (3) + (6) + (10) + ……..N times
ip: N=5 op: sum=35
by observing above program, the values of 'sum' variable in each cycle is 1, 1+2, 1+2+3, 1+2+3+4, etc
By adding all these values to some other variable(say 'sum2') we get final sum.

---

91) Program to print product of each term 1, 1*2, 1*2*3, 1*2*3*4, 1*2*3*4*5, …N times.
in mathematics, this sum is expressed as: 1!, 2!, 3!, 4!, 5!, 6!, …. N times
ip: enter N value : 7
op: 1, 2, 6, 24, 120, 720, 5040
note: Do not use nested-loop (using single loop we can solve it)

---

92) Program to find (1)+(1*2)+(1*2*3)+(1*2*3*4)+ (1*2*3*4*5) .....N times (1!+2!+3!+ …..N!)
1 + 2 + 6 + 24 + 120 + ……..N times
ip: N is 5 op: sum=153

---

93) Program to print product of each term 1, 1*2*3, 1*2*3*4*5,…N times.

ip: enter N value : 7

op: 1, 6, 120, 5040

note: do not use nested-loop and also do not use **'if-statement'** in the loop to check odd number.

logic: take 'p' to generate odd factorials, here multiply 'p' with (i+1)*(i+2) to get next fact value.

initially p=1, after multiplying 'p' with (i+1)*(i+2) at 1st cycle then 'p' becomes p=1*2*3→ p=3!

every time increment loop variable 'i' by 2 to get next odd number in the loop.

---

94) $X^1/1! + X^2/2! + X^3/3!$..... N times [do not use pow() fn]

ip: x=3, N=5

op: sum= 17.4 [3.0 + 4.5 + 4.5 + 3.375 + 2.025 → 17.4]

---

95) $X^1/1! - X^3/3! + X^5/5! - X^7/7!$ ..... 5 times. [sine series , this is very very important program]

ip: x=3, N=5

op: sum=0.1453 [(3.0) + (-4.5) + (2.025) + (-0.4339) + (0.05424)]

---

96) Write a program to count number of digits in a given number

ip: 29431 op: count=5

---

97) Write a program to print first digit of a given number.

ip: 2345 ip: 456

op: 2 op: 4

logic: repeatedly divide N=N/10 until N>9, finally N contains first digit.

if N>9, which means, N contained more than one digit.

---

98) Write a program to print sum of first and last digits of a given number.

ip: 2345 ip: 43 ip: 7

op: sum=7 (2+5) op: 7 (4+3) op: 7

step1: assign last digit to variable 'sum' [sum=n%10]

step2: now remove all digits except first digit from N, like above said.

step3: at this moment 'N' contains first digit, now add N to 'sum'

step4: print(sum)

---

99) Write a program to print 4-digit number in English words.

ip: N=3985

op: three nine eight five

step1: take 'd' with 1000, because N is 4-digit number

step2: divide N with 'd', and get the quotient eg: q=N/d → q=3985/1000 → q=3

now print 'q' in English words by substituting in following code

```
if(q==0) print("zero");
else if(q==1) print("one");
else if(q==2) print("two");
…
```

step3: remove first digit 3 from 3985, for that N=N%d; // because printing '3' is over

now N becomes 985

step4: decrement d to 100 (d=d/10), because N has 3-digits now.

step5: repeat these steps until d>0

---

100) Write a program to print given number N in English words.
(extension to above program, here N may have any number of digits)

ip: 2345 ip: 415
op: two three four five op: four one five

step1: take 'd' and generate its value to 10^{C-1}, where 'C' is no.of digits in N.
if N=123, then d=100
if N=4567 then d=1000
so the code to generate 'd' value is

```
d=1;
while(N/d>9)
{   d=d*10;
}
```

step2: now take one more loop, and extract digit by digit in N from left to right until N>0
for that divide N with 'd' and collect the quotient. Eg: q=N/d [q=2345/1000 → q=2]
step3: now print 'q' in English words
step4: remove first digit from N, for that divide N with 'd' and collect the remainder to N itself.
N=N%d [N=2345%1000 → N=345]
step5: down the d value to d=d/10; because N contains 3-digits now
step6: repeat step-2 to step-5 until N>0

---

101) Write a program to print right most odd digit in a given number, if odd digit not found then print the message "odd not found".

ip: 234**5**6 ip: 2468
op: 5 op: odd digit not found

step1: take 'X' to store odd number (initially take 'X' with -1)
step2: now check each digit of N
step3: if digit is odd then insert into X and stop the loop.
step4: if digit is not odd then do N=N/10 and check for next digit. (goto step2)
step5: after loop, if X==-1 then say "no odd found" or else print(X)

---

102) Write a program to print left most odd digit in a given number, if no odd is digit exist then print message "no odd digit found".

ip: 2**3**451 ip: 2468
op: 3 op: odd digit not found

step1: take 'X' to store odd number (initially take 'X' with -1)
step2: now check each digit of N
step3: if digit is odd then insert into X. (This may not be final odd, it may replace with next odd digit)
step4: now do N=N/10 and check for next digit. (goto step2)
step5: after loop, if X==-1 then say "no odd found" or else print(X)

---

103) code to scan **'N'** as input, and remove all odd digits in the number, and also check output N>500 or not?

ip: 23456 ip: 35 ip: 88341 ip: 7854
op: 246 (< 500) op: 0 (< 500) op: 884 (>500) op: 84(<500)

---

104) Write a program to scan '**N'** as input, and rearrange all odd digits to front and all even digits to rear. and also check whether output N is above 1000 or not? (take 2 loops, first loop for odd, second for evens)

ip: 23456 ip: 25 ip: 311 ip: 7854

op: 35246 (>1000) op: 52 (<1000) op: 311 (<1000) op: 7584 (>1000)

---

105) Program to print last Fibonacci number which is fall below N. Here N is input.

ip: 12 ip: 8 ip: 13 ip: 35

op: 8 op: 8 op: 13 op: 34

---

106) Write a program to print Fibonacci series values which are in between two given limits.

ip: n=10 150

op: ~~0, 1, 1, 2, 3, 5, 8,~~ 13, 21, 34, 55, 84, 139, ~~223, 362, 585~~

---

107) Program to print given number is in Fibonacci series or not?

ip: n=13 ip: 14

op: yes, it is in series op: no, not in series

---

108) Program to print prime factors of given N. (The product of factors should be equal to N)

ip: N=100

op: 2 2 5 5

step1: divide N with first factor 2, the number 2 is prime.
if N is divided with 2 then print(2) as prime factor and decrement N to N/2.

step2: repeat step1 as long as '2' divides the N.

step3: Now take 3 and proceed as long as 3 divides the N, as said in step1. Of course '3' is also prime.

step4: Now take 4, we know 4 is not prime, but 4 will not be divided the N because we already did with 2 before, so there should not be 2 multiples left behind in N. [you may ask one question, why to divide with 4 when it is not prime, because it is difficult to take only primes, taking only primes is another problem. So continuously/blindly divide the N with 2,3,4,5,6,7,8,9 ….]

step5: repeat this process until N>1

Let us see how N value changes in the loop

N →	100	50	25	5	1
i →	2	2	~~2,3,4~~, 5	5	Stop

---

109) Program to print prime factors of given N. The process is same as above program but don't repeat factors more than once. (not like 2, 2, 5, 5)

ip: N=100 op: 2 5

logic: Here take extra variable 'prev' to store previous value of 'i' in the loop(for first time, take prev=1)

```
if(n%i==0)
{       if(prev!=i)     // if previous printed factor is not equal to current factor then print
        {       printf(" i as factor ");
                prev=i;     //  take this current 'i' value as 'prev'  for next cycle
        }
        n=n/i ;
}
```

---

110) Program to accept a number N from keyboard and find whether it has perfect square root or not?

ip: 16 ip: 15

op: yes (4^2) op: no

logic: 1) Repeat the loop until i^2<N where i=1, 2, 3, 4, 5,…

2) after completion of loop, if i^2 == N then say "yes" or else "no"

the looping is as follows

if N=16 then

while(1*1<16) is true

while(2*2<16) is true

while(3*3<16) is true

while(4*4<16) is false

now 4*4==16 is true, so print "yes"

if N=15 then

while(1*1<15) is true

while(2*2<15) is true

while(3*3<15) is true

while(4*4<15) is false

now 4*4==15 is false, so print "no"

---

111) Program to accept a number N through keyboard and find whether it is power of 2 or not?

ip: 8 ip: 18

op: yes (2^3==8) op: no

logic1: Repeatedly Compare N with 2^0, 2^1, 2^2, 2^3, 2^4, 2^5 … Until 2^i<N

Finally, after loop if N==2^i then say "yes", if not then say "no".

logic2: 1) cut down N=N/2 as long as N is even

2) finally, after the loop, if N==1 then say "yes", or else, say "no"

for example, if N=20 then 20, 10, 5 (here N==1 is false so print "no")

for example, if N=16 then 16, 8, 4, 2, 1 (here N==1 is true so print "yes ")

---

112) Write a program to find LCM of 3 numbers

ip: 20 15 35

op: 420 (2*2*3*5*7)

1. Let x, y, z are three input numbers
2. Start finding LCM of three with first factor 2 (i=2)
3. Now divide each x, y, z with 2. If any x, y, z is divided then take 2 as one multiple in LCM. (LCM=LCM*2) also decrement divisible numbers to quotient obtained in the division (if X%2==0) X=X/2.
5. Now repeat step-3, step-4 as long as 2 divide any of x, y, z
6. if 2 no longer divided, then next try with next factors 3, after 4, 5…etc, (i++)

repeat this process until any of these (x, y, z) > 1. for example, while(x>1 || y>1 || z>1) {….}

Let the numbers are 20, 15, 35 and following table shows how …

2	20	15	35
2	10	15	35
~~2,~~ 3	5	15	35
~~3,4,~~5	5	5	35
~~5,6,~~7	1	1	7
	1	1	1

---

113) Write a program to find square root of a given number

Use Babylonian method of guess and divide, and it is truly faster. (Scientist name)

It is also the same as you would get applying Newton's method.

See for an example, how to find it

The square root of 20 using 10 as the initial guess (n/2)

Guess	Divide	Find average
• 10	20/10 = 2	average 10 and 2 to give new guess of 6
• 6	20/6 = 3.333	average 3.333 and 6 gives 4.6666
• 4.666	20/4.666= 4.1414	average 4.666, 4.1414= 4.4048
• 4.4048	20/4.4048=4.5454	average = 4.4700
• 4.4700	20/4.4700=4.4742	average = 4.4721

repeat this process until previous & current guess values are same in the looping.

---

114) Write a program to find hexadecimal number(N) from a given binary number

ip: 370359 ip: 159

op : 5A6B7 op: 9F

Repeatedly divide the N with 16, and collect(add) the remainders into variable **'sum'.** The steps are

1.Rem=N%16

2.Add Rem to sum, this is like sum=sum*100+Rem

3.cut down N to N/16

4.Repeat these steps until N>0

Let N=370359

16	370359		sum ↓
16	23147 → 7	0*100+7	→ 7
16	1446 → 11 (B)	7*100+11	→ 711
16	90 → 6	711*100+6	→ 71106
16	5 → 10(A)	71106*100+10	→ 7110610
	→ 5	7110610*100+5	→ 07 11 06 10 05

The hexadecimal value collected in '**sum'** as → 07 11 06 10 05 (7B6A5)

but output should be displayed as → 5A6B7 (extract 2-digits at a time right-to-left from **'sum'** and print)

use two loops, first loop to generate **'sum'** and second loop is to print in hexadecimal form.

the second loop as

```
while(sum>0)
{       rem=sum%100;
        if(rem<10) printf("%d", rem);
        else printf("%c", 'A'+rem%10) ;          // 10 for A, 11 for B, 12 for C..
        sum=sum/100;
}
```

---

115) Write a program to find second big digit in a given number.

ip: 2751 ip: 5555

op: 5 op: 5

first find big and then use following logic

```
while(n>0)
{    r=n%10;     n=n/10;
     if( sbig==big && r<big || r>sbig && r<big)  sbig=r;
}
```

---

116) Wri0te a menu driven program to find given number is odd/even, Palindrome, Prime, Armstrong, and perfect or not; when we execute the program, the menu appeared as given below

```
Menu run
==========================
1. Even/odd
2. Palindrome
3. Prime or not
4. Armstrong
5. Perfect
0. exit
Enter choice [1,2,3,4,5,0]: 1
```

```
scanf("%d",&N);
while(1)
{   printf("\n1.Even/odd   \
        \n2.Palindrome or not \
        \n3.Prime or not  \
        \n4.Armstrong or not  \
        \n5.Perfect or not  \
        \n0.exit  \
        \n\t Enter choice :");
    scanf("%d", &ch);
    switch(ch)
    {     case 1:   if(n%2==0) printf("even");
                    else printf("odd");
                    break;
          case 2:   ------
                    break;
          ------
          ------
          case 0: exit(0);
    }
}
```

Nested Loops

1)
```
for( i=1;  i<=5;  i++ )
{     printf(" A ");
      for( j=1;  j<=3;  j++ )
            printf( " B " );
      printf(" C \n " );
}
```

2)
```
for( i=1;  i<=5;  i++ )
{     printf(" A ");
      for( j=1;  j<=3;  j++ )
            printf(" %d  " , j );
      printf(" B \n " );
}
```

3)
```
for( i=1;  i<=5;  i++ )
{     printf(" A ");
      for( j=1;  j<=3;  j++ )
            printf( " %d  " , i );
      printf(" B \n " );
}
```

4)
```
for( i=1;  i<=5;  i++ )
{     for( j=3;  j<=8;  j++ )
            printf( " %d  " , j );
      printf(" \n " );
}
```

5)
```
for( i=1;  i<=5;  i++ )
{   for( j=1;  j<=3;  j++ )
          printf("\n %d :: %d", i , j );
}
```

6)
```
for(i=1; i<=5; i++)
{     printf("\n table %d :: ", i);
      for( j=1;  j<=10;  j++)
            printf("%d  ",  i*j );
}
```

7)
```
for(i=8;  i>=1;  i-- )
{   for( j=1;  j<=i;  j++)
          printf("%d  " , j );
     printf("\n");
}
```

8)
```
for( i=1; i<=8 ; i++)
{     for( j=i;  j<=8;  j++)
            printf("%d  ",  j );
      printf("\n");
}
```

9)
```
for( i=4;  i<=10;  i++)
{     for( j=1;  j<=i ;  j++)
          printf("%d  ",  j );
      printf(" \n ");
}
```

10)
```
for(i=1;   i<=8 ;  i++ )
{     for( j=1;  j<=5;  j++ )
          printf("%d  ",  i );
      printf(" \n ");
}
```

11)
```
for( i=7;  i>=1 ;  i-- )
 {     for(j=i;  j<=9;  j++)
          printf("%d  ",  j );
       printf("\n");
 }
```

12)
```
for(i=7;   i>=1;  i-- )
 {      for( j=9;  j>=i;  j-- )
            printf("%d  ",  j );
        printf("\n");
 }
```

13)
```
for( i=9;  i>=1; i-- )
{     for( j=i; j<=9;  j++)
          printf("%d  ",  j );
      printf("\n");
}
```

14)
```
for( i=9; i>=1; i--)
 {      for( j=9;  j>=i;  j--)
            printf("%d  ",  j );
        printf("\n");
 }
```

15)
```
for(i=1; i<=4; i++)
{   printf("\n %d  => " , i );
     for(j=i+1; j<=7; j++)
          printf("%d", j );
    printf("\n");
}
```

16)
```
for(i=1; i<=5; i++)
{   for(j=1; j<=5; j++)
          printf("%d",  i  );
     printf("\n");
}
```

17)
```
for( i=1;  i<=5;  i++)
{       for( j=1;  j<=i;  j++ )
                printf(" * ");
        printf("\n");
}
```

18)
```
n=2835;
while(n>0)
{       r=n%10;
        for(j=1; j<=r; j++)
                printf("*");
        printf("\n");
        n=n/10;
}
```

19)
```
for(i=1; i<=5; i++)
{    for(j=1; j<=5; j++)
            printf("%d",   j  );
      for(j=5; j>=1; j--)
            printf("%d",   j  );
      printf("\n");
}
```

20)
```
for(i=1; i<=5; i++)
{     for(j=1; j<=i; j++)
            printf("%d",   j  );
      for(j=i; j>=1; j--)
            printf("%d",   j  );
      printf("\n");
}
```

21)
```
for(i=1; i<40; i++)
 {    printf("%d",   i  );
      if( i%5==0)
            printf("\n");
 }
```

22)
```
for(i=5; i<=10; i++)
{       for(j=i;  j<=i+5; j++)
            printf("%d ", j);
        printf("\n");
}
```

23)
```
for(k=8, i=1;  i<=5;  i++ )
{     for(j=1;  j<=k;  j++)
            printf("%d  ",  j );
      printf("\n");
      k--;
}
```

24)
```
for(k=8, i=1;  i<=5;  i++, k-- )
{       for(j=k;  j>=1;  j--)
            printf("%d  ",  j );
        printf("\n");
}
```

25)
```
for(i=1;  i<=5;  i++ )
{     for(j=1;  j<=6-i;  j++)
            printf("%d  ",  j );
      printf("\n");
}
```

26)
```
for(i=1;  i<=3;  i++ )
{     for(j=1,k=10;   j<=k;   j++,k--)
            printf("%d  ",  j );
      printf("\n");
}
```

27)
```
for(k=i=1;  i<=5;  i++ )
{     for(j=1;  j<= i;  j++)
            printf("%d  ",  k++);
      printf("\n");
}
```

28)
```
for( i=9; i>0; i--)
{    for(j=3;  j<i; j++)
            printf("%d  " , j );
     printf("\n");
}
```

29)

30)

Nested Loops

1) produce the following pattern output

```
3456789
3456789
3456789
-----------
8 rows
```

2)

```
9876543
9876543
9876543
-----------
8 rows
```

3)

```
12345678
2345678
345678
 ---
78
8
```

4)

```
12345678
1234567
123456
----
12
1
```

5)

```
987654321
98765432
9876543
987654
-------
```

6)

```
987654321
87654321
7654321
654321
-------
```

7)

```
9
98
987
9876
98765
-----------
987654321
```

8)

```
9
89
789
6789
56789
----------
123456789
```

9)

```
1
12
123
1234
12345
----------
8 rows
```

10)

```
1
21
321
4321
54321
-------
8 rows
```

11)

```
1111111
2222222
3333333
4444444
-----------
8 rows
```

12

```
8888888
7777777
6666666
5555555
-----------
1111111
```

13)
```
1
22
333
4444
55555
---------
8 rows
```

14)
```
88888888
7777777
666666
55555
-------
22
1
```

15)
```
1234554321
1234554321
1234554321
1234554321
1234554321
----------
8 rows
```

16)
```
1234567887654321
12345677654321
123456654321
12345554321
12344321
123321
1221
11
```

17)
```
11
1221
123321
12344321
1234554321
123456654321
--------------
8 rows
```

18)
```
1
121
12321
1234321
123454321
12345654321
------------
8 rows
```

19)
```
*                    // printf("*");
**
***
****
---------
8 rows
```

20)
```
ip:  4315
op:  *****
     *
     ***
     ****
```

21)
```
    1
   2 2
  3 3 3
 4 4 4 4
5 5 5 5 5
```

4 blanks →					1				
3 blanks→				2		2			
2 blanks→			3		3		3		
1 blanks→		4		4		4		4	
0 blanks→	5		5		5		5		5

Solution is in next column ➔

```
for(i=1; i<=5; i++)
{       for(j=1; j<=5-i; j++)
                printf("␣");       // let ␣ is a space-bar
        for(j=1; j<=i; j++)
                printf("%d␣", i);
        printf("\n");
}
```

22)
```
5 5 5 5 5
 4 4 4 4
  3 3 3
   2 2
    1
```

let us see how to print spaces before numbers

0 blanks →	5		5		5		5		5
1 blanks→		4		4		4		4	
2 blanks→			3		3		3		
3 blanks→				2		2			
4 blanks→					1				

23A) Write a program to print multiplication table of a given N.
23B) Write a program to print multiplication tables from 3 to 19 and each table with 10 terms

24) Write a program to print multiplication tables from 3 to 19 by skipping 5, 10 and 11 tables
[use **continue** statement]

25A) Write a program to print factorial of a given input N. (if N=5 then factorial is 5*4*3*2*1→120)
25B) Write a program to print sum of factorials of each digit in a given number
ip: 245
op: 2!+4!+5!=>2+24+120=>146

26A) Write a program to accept N as input, and print whether it is palindrome or not?
26B) Write a program to print palindrome numbers in between 361 to 765
op: 363, 373, 383, 393, 404, 414,424, 434, 444, …etc

27A) Write a program to accept N as input and print whether it is prime or not?
27B) Write a program to print list of primes between 10 to 100
op: 11, 13, 17, 19, …97

28) Write a program to print twin-prime numbers from 2 to 100
Twin means 3-5, 5-7, 11-13, … (difference is 2)

```
previous=2;     // let us take as first prime
for(N=3;  N<100;  N=N+2)     //  remember, only odds are primes
{       here  check 'N', whether it is prime or not?       // use bool logic
        if(bool==1)
        {     if(N - previous==2)
                  print(previous, N  as twins)
              previous=N;
        }
}
```

29) Write a program to accept a number and add up all digits until the number gets single digit;
for example
19999=>1+9+9+9+9=>37
37=>3+7=>10
10=>1+0=>1

```
while(N>9 )     //  if N has more than one digit
{    here take sum to zero
     here take inner-loop and add all digits of N to sum
     after inner-loop, move 'sum' value to 'N'  and continue the outer loop
}
finally N contains single digit, so print it here.
```

Home Work

30)
```
1   2   3   4   5
6   7   8   9   10
11  12  13  14  15
16  17  18  19  20
------
8 rows
```

31)
```
1
2  3
4  5  6
7  8  9  10
-----
8 rows
```

31A)
```
1
3  2
6  5  4
10 9  8  7
-----
8 rows
```

32)
```
    1
   222
  33333
 4444444
555555555
```

33)
```
555555555
 4444444
  33333
   222
    1
```

34)
```
555555555
 4444444
  33333
   222
    1
   222
  33333
 4444444
555555555
```

35)
```
    1
   222
  33333
 4444444
555555555
 4444444
  33333
   222
    1
```

36) use three nested loops.

Output:
```
123
132
213
231
321
312
```

37)
```
1
01
010
1010
10101
----------
8 rows
```

38)
```
1   2   3   4   5       // row1
10  9   8   7   6

11  12  13  14  15      // row2
20  19  18  17  16

21  22  23  24  25      // row3
30  29  28  27  26
```

39)

ip: 5263

op:
```
*****
**
******
***
```

40) Pascal triangle
```
      1
    1   1
  1   2   1
1   3   3   1
1 4   6   4   1
1 5  10  10 5  1
1 6 15 20 15 6 1
```

41) ip: n=5
```
1 1 1 1 1
2       2
3       3
4       4
5 5 5 5 5
```

42)code to print previous prime of a given number N (here N>2).

ip: 10 ip: 100 ip: 19
op: 7 op: 97 op: 17

Note: start finding primeness from N-1 to 2 (in reverse order)

43) Write a program to print next prime of a given number N.

ip: 13 ip: 100 ip: 17
op: 17 op: 101 op: 19

Note: start finding prime from N+1 to infinate

44) The prime factors of 13195 are 5, 7, 13 and 29.
What is the largest prime factor of the number 50001?

- remember the largest factors exist from N/2 to 1(in reverse order), so take loop as for i=n/2 to 1
- if 'i' is factor and it is prime then stop the loop and print(i)
- Remember that. only odd numbers can be a prime (not even numbers)

45) A palindromic number reads the same both ways. The largest palindrome made from the product of two 2-digit numbers is 9009 = 91 × 99.
Find the largest palindrome made from the product of two 3-digit numbers.

46) A Pythagorean triplet is a set of three natural numbers, a < b < c, for which, $a^2 + b^2 = c^2$
For example, $3^2 + 4^2 = 9 + 16 = 25 = 5^2$. ($3^2 + 4^2 = 5^2$)
There exists exactly one Pythagorean triplet for which a + b + c = 1000. Find the product a,b,c.

1D-Arrays

---

1) Code to accept 5 values to array and find sum of all odd numbers

ip: 4 7 10 11 5

op: 7+11+5 → 23

```
void main()
{       int  a[5] , sum, i ;
        printf("enter 5 values:");
        for( i=0;  i<5; i++)                // this  loop  for  scanning  5  values
        {       scanf( "%d", &a[i] ) ;
        }
        for( i=0;  i<5; i++)                // this  loop  for  adding odd values
        {       if  a[i]  is  odd  then add  a[i]  to  'sum'
        }
        print  sum  of  adds  here
}
```

---

2) Code to accept 5 values to array and convert all -ve values to +ve values if any

ip: 4 6 -13 11 -5

op: 4 6 13 11 5

---

3) Code to accept 5 values to array and convert all -ve values to +ve values and vice-versa

ip: 4 6 -13 11 -5

op: -4 -6 13 -11 5

---

4) Code to accept 5 values to array and increment even numbers to next odd in the array (like a[i]++)

ip: 4 7 12 17 10

op: 5 7 13 17 11

---

5) Code to accept 5 values to array and count number of 3 divisible in the array.

ip: 4 6 11 12 5 ip: 4 16 13 11 5

op: count=2 op: count=0

---

6) Code to accept 5 values and print first odd from left to right.

ip: 4 7 10 11 4 ip: 4 8 10 12 14

op: 7 op: no odd found

---

7) Code to accept 5 values and print second odd from left to right.

ip: 4 7 10 11 4 ip: 4 8 11 12 14

op: 11 op: no second odd found

---

8) Code to accept 5 values and print sum of left most and right most odd sums.

ip: 4 7 13 11 8 ip: 4 8 11 12 14 ip: 10 12 14 16 18

op: 7+11 op: 11+0 → 11 op: 0+0 → 0

---

9) Code to accept 5 values to array and find at least one value is –ve or not?

ip: 4 6 -13 11 -5 ip: 4 6 13 11

op: yes, -ve exist op: no, -ve not exist

Logic: if any value a[i]<0 then –ve exist, or else not. ----

---

10) Code to accept 5 values from K.B and find any two adjacent pair of elements are same or not?

ip: 4 7 11 12 13 ip: 4 7 **11 11** 13

op: no pair is found op: yes pair is found // if(a[i] == a[i+1])

---

11) Code to read 5 values to array, and find whether they are in ascending or not?

ip: 12 15 19 22 31 ip: 12 15 22 19 31

op: yes, in ascending order op: no, not in ascending order

Logic: compare a[i] with a[i+1] for all i=0,1,2,3. If any a[i]>a[i+1] then not in ascending order.

---

12) Code to search an element '11' is exist or not? Let array contained 5 values scanned from KB. Also print its index value(position in the array).

ip: 4 6 11 12 5 ip: 4 16 13 12 5

op: 11 exist, its position is 3 op: 11 not exist

---

13) Complete the following code to print no.of days in a given month.

here array pre initialized with all month values (ignoring leap year)

ip: 4 ip: 2 ip: 5

op: 30 days op: 28 days op: 31 days

```
void main()
{   int a[13]={0, 31, 28, 31, 30, 31,....};   // first value 0 is dummy, because month index start with 1
    printf("enter month in date:");
    scanf("%d", &m);
    // here substituting 'm' in a[ ] to get days
}
```

---

14) Code to accept 5 values from KB and print each number's multiples as shown below (use nested-loop)

ip: 14 16 13 11 24

op: 14 => 1, 2, 7, 14

16 => 1, 2, 4, 8, 16

---

15) Code to accept 5 values to array and print only perfect number

ip: 6 17 28 31 1

op: the perfect numbers are: 6, 28 (if sum of all divisible is equal to given number then it is perfect)

---

16) Code to accept 5 values to array and print reverse of each number (use nested loop)

ip: 123 21 529 1312 65781

op: 321 12 925 2131 18756

---

17) Code to accept 5 values to array and print palindromes in the array.

ip: 123 121 529 1312 656

op: 121 656

---

18) Code to accept 5 values to array and print only primes in the array

ip: 11 17 21 31 15

op: the primes are: 11, 17, 31

---

19) Code to accept N values to array and print sum, average, and big of them.

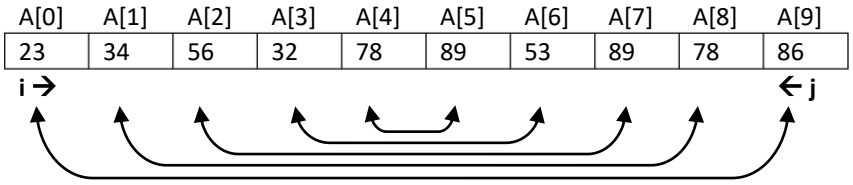
```
void main()
{   int a[20], N, sum, …. ;
    printf("enter how many input values:");
    scanf("%d", &N);
    if( N>20)
    {       printf("Error, array size not enough");
            return;
    }
    for( i=0;  i<N; i++)
    {       printf("enter value of a[%d] :", i );
            scanf("%d", &a[i] ) ;
    }
    ----
}
```

```
running of program
input
~~~~~~~~~~~~~
enter how many input values: 6
enter value of a[0]: 7
enter value of a[1]: 10
enter value of a[2]: 14
enter value of a[3]: 23
enter value of a[4]: 16
enter value of a[5]: 18
output
~~~~~~~~~~~~~
sum is : 88
average is : 14.66
big is : 23
```

---

20) Code to accept N numbers from keyboard and print all elements after reversing them.

To reverse all elements, swap elements with opposite sides i.e., swap the first element with the last element, second element with the previous of last, and so on. (Note: Nested loop not required)



Take 'i , j' as loop variables, 'i' for forward direction and 'j' for backward direction and swap every pair of a[i],a[j] until i<j. The loop is as follows: for(i=0, j=n-1; i<j; i++, j--) { swap a[i] , a[j] }

---

21) Let array a[ ] contained N elements, now find whether array is symmetric or not?

Symmetric means: The values are same if we read in any direction (left→right or right→left)

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	A[6]
51	21	32	61	32	21	51

← This array is symmentric

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	A[6]
11	**99**	32	61	32	**21**	11

← This array is not symmetric

logic: compare first & last elements, second & previous of last, … until i<j as above shown.

---

22) Let N value has non repeated digits like 742 , now just print all digits in ascending order 247

step1: take array with size 10, and initialize all cells with 0. This is like: int a[10]={0};

step2: now take digits one by one from N, if digit is 2 then set a[2]=1, in this way fill array cells with 1.

step3: now print all array index values of 'i' where a[i]==1 , here i=0,1,2,3,…,9

if N is 742, let us see how array cells filled with 1's

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]
0	0	1	0	1	0	0	1	0	0

---

23) Let N value like 777233 (digits may repeated) , now print all digits in ascending 233777

step1: take array with size 10, and initialize all cells with 0,

step2: now take each digit from N, if digit is 3 then increment a[3]++, in this way increment array cells.

step3: now print array index value of 'i' when a[i]>0, (if a[i] value is 3 then print 'i' value 3 times)

ip: 777233

op: 233777

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]
0	0	1	2	0	0	0	3	0	0

---

24) Let N value like 777233 (digits may repeated) , now print highest repeated digit(7)

ip: 777233 op: 7

---

25) Code to scan values one by one until last input is -1, the input values must lie in between 0 to 9, if not then say error message "invalid-input" and skip such number (do not take as input value). Finally, print how many times each value repeated.

input: 2↵ 8↵ 5↵ 2↵ 5↵ 5↵ 2↵ 8↵ 2↵ 2↵ -1↵ (-1 to stop)

output: 2 repeated 5 times
5 repeated 3 times
8 repeated 2 times

1. First take array A[] with size 10 and initialize all cells with zero. This is like: **int A[10]={0};**
2. If the input N is 2 then increment A[2]++ (Here each cell A[i] is like a count variable)
3. If the input is 5 then increment A[5], in this way increment array cells.
4. Finally, print all nonzero cell values how many times each repeated

let us see following picture, how cells will be incremented for above input values.

a[0]	a[1]	[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
0	0	5	0	0	3	0	0	2	0

```
void main()
{       int  a[10]={0},n, i ;
        while(1)
        {       printf("enter a value ( 0 to 9 )");
                scanf("%d", &n);
                if(n==-1) break;
                if( n<0 || n>9 )
                {   printf("invalid input(0-9), try again ");  continue;  }
                    a[n]++;
                    ------
```

---

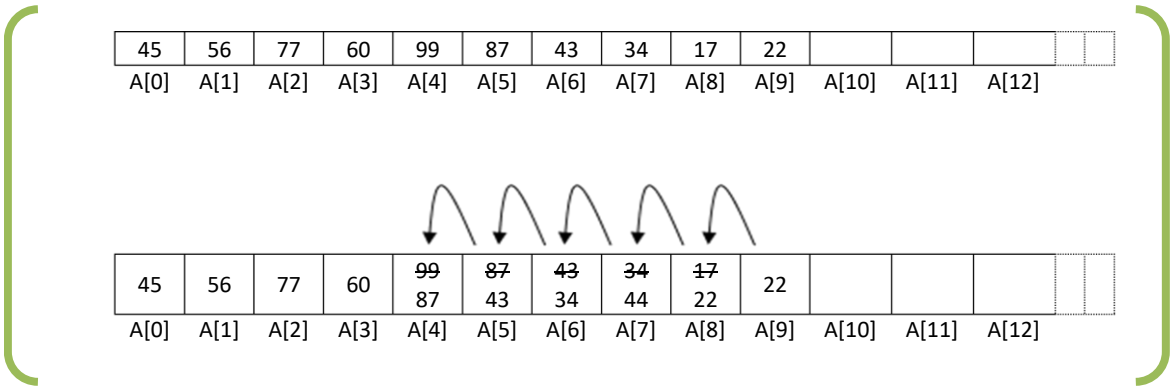
26) Deleting kth element in the array

✦ once array is created, physically we can't delete some cells in the array, alternative is, delete the data by shifting elements. Here to delete kth element, replace a[k] with a[k+1], a[k+1] with a[k+2], …etc. so shift all elements one position back from a[k]. Following picture shows how to delete it.

✦ the following array contained 10 values, and it gives demo how to delete 5th position element.

✦ The following code deletes 5th position element in the array (shifting to previous positions).

```
for(i=5; i<10; i++)
    a[i-1] = a[i];      //  it replaces  a[4] by a[5],   a[5] by a[6], …etc.
```



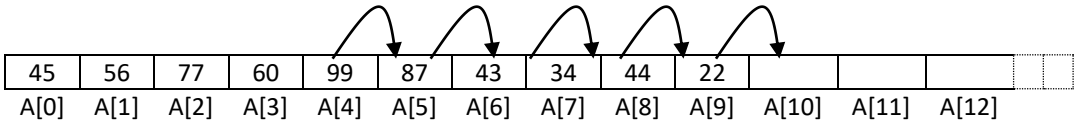
Now write a program, where accept N values from keyboard and delete kth element in the array (k<N) later print all elements after deleting kth element.

---

27) Inserting a new element at kth position

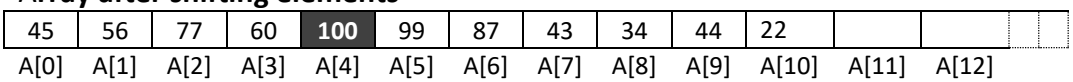
Code to insert a new element at kth position in an existing array of N elements where k<N

To insert a new element at A[k], shift all existing elements to right side by one position, so that we get a gap at A[k], where new element can be inserted. Following picture shows how to shift elements.



For example, to insert a new element 100 at A[4], shift all elements 22, 44, 34, 43, 87, 99 to the right side by one position, so that we get a gap at 99, **where** 100 can be inserted.

Array after shifting elements



Logic is as follows:

```
printf("enter new element and its position :");
scanf("%d%d",  &new ,  &pos );
for(i=n;  i>=pos  i--)     // shifting elements to right side
{      a[i]=a[i-1];
}
a[i]=new;                  // inserting new element  at a[pos]
n++;                       // as one new element inserted, so increase count.
```

Home Work

28) In the following example, the list of 0 to 10 factorials already calculated and stored in the array, now our job is to scan N,R values and print NcR value. NcR → N! / ((N-R)! * R!)

ip: 4 , 2 ip: 7 , 1

op: 6 op: 7

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	A[9]	a[10]
1	1	2	6	24	120	720	5040	---	---	---

```
void main()
{  int a[10]={ 1, 1, 2, 6, 24, 120, 720, 5040 } ;   // pre-initialization of array with factorial values
   int N, R;
   scan( N, R)
   here substitute N,R values in the array to get NcR value and then print result.
}
```

---

29) Write a program to check given date is valid or not

step1: take array with month-values (pre initialization of array)

```
int  arr[13]={0, 31, 28, 31, 30, 31,....};          // first value is zero
```

step2: scan date (d, m, y)

step3: here update February month from 28days to 29days for leap-year.

```
arr[2]=28+(y%4==0);          // if year is leap-year then  y%4==0  gives 1 or else 0
```

step4: now check the date is valid or not?

```
if( m<1 || m>12  || d<1 || d>arr[m] )
    printf("valid date");
else
    printf("invalid date");
```

---

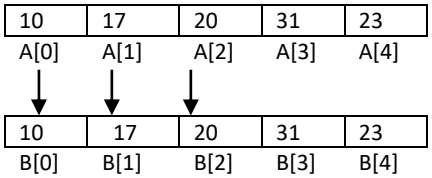
30) Let two arrays have same number of elements with same order, now prove by comparing two array elements are same or not as per values and order. Here scan N values to both arrays.

ip: a[ ] = {10, 17, 20, 31, 23 }; ip: a[ ] = {10, 17, 20, 31, 23};

b[ ] = {10, 17, 20, 31, 23 }; b[ ] = {10, 17, 20, 23, 31};

op: yes, both are equal op: no, both are not equal

the comparison as given below figure



---

31) print fibonacci numbers in reverse order.

ip: n is 10 (10 number to be printed)

op: 34 21 13 8 5 3 2 1 1 0

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]
0	1	2	3	5	8	13	21	34

Step1: take array, and generate Fibonacci numbers and store into array

```
initially x=0, y=1 //first two terms of series
for(i=0; i<n; i++)
{   a[i]=x;  new=x+y;  x=y; y=new };
```

step2: now print back to front of array.

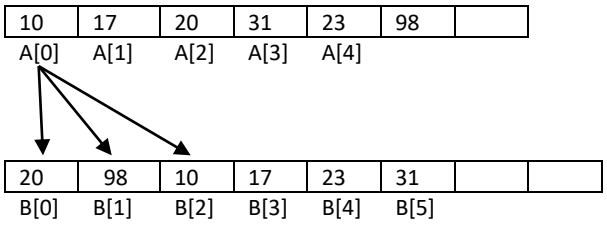
---

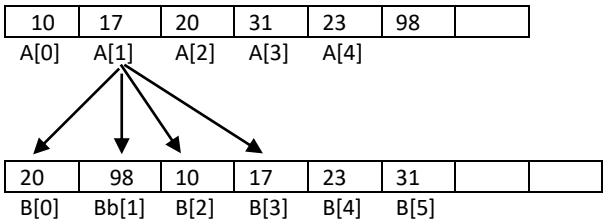
32) Let two arrays A[], B[] have N elements each, now find how many common elements found in arrays.
If any element in A[] is also exist in B[] then it is said to be common element.
(Let us take both arrays have same count of input values without duplicate in the same array)

ip: a[ ] = {10, 17, 20, 31, 23, 98 };
b[ ] = {17, 20, 10, 18, 22, 32 };
op: count=3

ip: a[ ] = {10, 17, 20, 31, 23, 98};
b[ ] = {17, 29, 13, 98, 23, 31};
op: count=2

the comparison is as given below figure





---

33) Let array contained N values, in which some elements may repeated more than once, anyway print only unique elements(not repeated elements).

ip: 14 16 18 20 14 38 21 16 14 14
op: 18 20 38 21

---

34) Write a program to accept two array of elements (two sets) and each contained N1, N2 values respectively, let the array names are A[], B[]. now print

1) print A[] Ω B[] (A intersection B , here print common elements of A , B)
2) print A[] – B[] (print elements which are in A but not in B)
3) print A[] U B[] (A union B) → A[] + (B[]-A[])

note: Write 3 separate programs for each output

---

35) Code to accept N values to array (here use A[0] as N, and store actual input values of array from A[1]) and print only odd values in the array.

```
int A[100], i ;
printf("enter no.of input values :");
scanf("%d",  &A[0]  );
printf("enter %d values to array :", A[0] );
for( i=1;  i<=A[0];  i++)
{   scanf("%d",  &A[i] );
}
-----
```

---

36) For the following set of sample data, compute the standard deviation(sd) and the mean.

ip: A[] = { 7 , 1 , -6 , -2 }

op: result=4.74

The formula for standard deviation is: $sqrt(\sum(A[i]–M)^2/N)$

Here mean **M** is nothing but average of all values, the formula as given below

$sd = sqrt ((A_0-M)^2 + (A_1-M)^2 + (A_2-M)^2 + (A_3-M)^2 + \dots + (A_n-M)^2 / N)$

---

37) Code to accept N values from keyboard and count pair of adjacent elements

ip: 14, 10, 9, 10, 10, 8, 8, 8, 11, 10, 17, 17, 17, 17, 17, 20.

op: count=4.

note: if 3 pair of elements found in adjacent place then take them as one pair(not two pairs), for example 8.

the code is like:

```
for( i=0 ; i< n-1 ; i++ )
    if( a[i] == a[i+1] )
    {   count++;
        i++;        // this extra i++ is to skip next pair value
    }
```

---

38) Code to accept 'N' values into array, later remove all occurrences of 8 in the array

ip: 14, 10, 9, 10, 8, 8, 11, 10, 8, 17, 20 8.

op: 14, 10, 9, 10, 11, 10, 17 20.

---

39) code to scan 5 odd values, while scanning, if user entered input is even then skip such value, if odd then insert into array, finally print how many primes exist in this odd values.

ip: 11 ↵ (inserted)
10 ↵ (skipped)
17 ↵ (inserted)
……

op: the primes are: 11, 17, 31

eg: // below logic scans 5 odd values.

```
for( i=0;  i<5;   )
{   scan(N);
    if( N%2==1)
    {     a[ i ]=N;      //  inserting N value into array
          i++;           //  as one odd value inserted into array, so incrementing 'i' by 1 .
    }
    else   printf("even input, skipped\n");
}
```

here write code to check and print all primes in the array.

---

40) Let array a[] pre initialized with 8 values (fixed values) , now print sum of 4 continuous values from given input 'k'. eg: sum of a[k]+a[k+1]+a[k+2]+a[k+3]. Use round robin method if a[i] exceeds array size.

if k=1, then a[1]+a[2]+a[3]+a[4]

if k=6, then a[6]+a[7]+a[0]+a[1] // last index is round robin because there is no a[8] , a[9]

if k=7, then a[7]+a[0]+a[1]+a[2]

* round Rabin means circular direction, when we reach to end point, then restart from zero.

```
void main()
{   int a[8]={ 11,22,33,44,55,66,77,88};     // pre-initialization of array  with 8 values.
    int  k;                  // the input value must be  0  to 7, if not then show an error.
    scanf(k)
    for( i=0;  i<4; i++)        // loop for addition of 4 values.
    {   sum=sum+a[k];
        k++;
        if(k==8) k=0;   or  k=k%8;
        -------
```

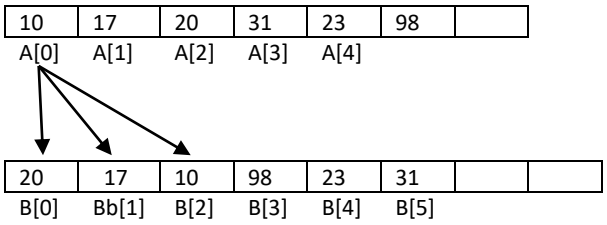
---

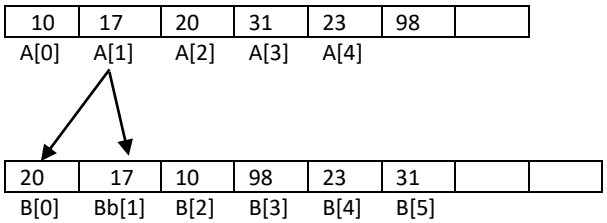
41) Let two arrays have N elements each, now find both arrays have same values or not (values may be in any order)
(Let us take both arrays have same count of N values without duplicate in the same array)

ip: a[ ] = {10, 17, 20, 31, 23, 98 };
b[ ] = {17, 20, 10, 98, 23, 31 };
op: equal

ip: a[ ] = {10, 17, 20, 31, 23, 98};
b[ ] = {17, 29, 13, 98, 23, 31};
op: not equal

the comparison as given below figure





---

42) Code to accept 'N' values into array, later remove all duplicates in the array (remove repeated values)

ip: 14, 10, 9, 10, 10, 8, 8, 8, 11, 10, 17, 17, 17, 17, 17, 20.
op: 14, 10, 9, 8, 11, 17, 20.

---

43) Let array a[ ] contained N values, count frequency of each number (how many times repeated)

ip: 14 16 18 20 14 38 21 16 14 14
op: 14 repeated 4 times
16 repeated 2 times

```
for(i=0; i<N; i++)              // this loop is to check all elements in the array
{       count=0;
        for(j=i; j<N; j++)      // this loop is to check repeated elements
            if(a[i]==a[j])      // compares a[i] with all other in the array
                count++;
        printf("\n %d repeated %d times", a[i], count);
}
```

The problem with this code is, the repeated values like 14 prints more than once. For example
when a[0] compared with a[4], a[8], a[9] then it prints output as "14 repeated 4 times"
but in the next iterations of loop, the 14 in a[4] is once again compared with remaining elements a[8],a[9].
To solve this problem, check a[i] with previous elements.
That is, before counting a[i] with remaining elements, first check a[i] with previous elements, if a[i] found in previous then skip it for counting.

---

44) Code to fill array with prime numbers from 2 to 1000.

The filled array output as: a[ ] = {2,3,5,7,11,13,17,19, …};

Hint: for example, to find prime-ness of 'N=35', then check by dividing with previous primes which are already explored by previous iterations of loop, for example 2, 3, 5, 7, 11,…<=35/2

Initialize array with first prime 2 and start loop from 3.

```
void main()
{   int a[100]={2} , count=1;   // first prime(2) is initialized with array so take count with 1.
    for(N=3; N<100; N++)
    {   bool=1;
        for( i=0; a[i]<=N/2; i++)   // taking previous primes like a[0],a[1],a[2] …
        {  -----
        }
        if( bool==1)                // if prime inserting into array
            a[count++]=N;
    }
    -----
}  this is fastest technique to check prime-ness of one number with previous primes.
```

---

imp 45) Program to accept two polynomials and find addition and multiplication of them.

For example, the polynomials represented using arrays as given below.

$f(x) = 7x^5 + 4x^3+2x+9$

In array representation, the array index itself is taken as exponent of polynomial whereas values in the array taken as coefficients . This is as given picture

```
int a[10];     // say, maximum degree of polynomial is 9
```

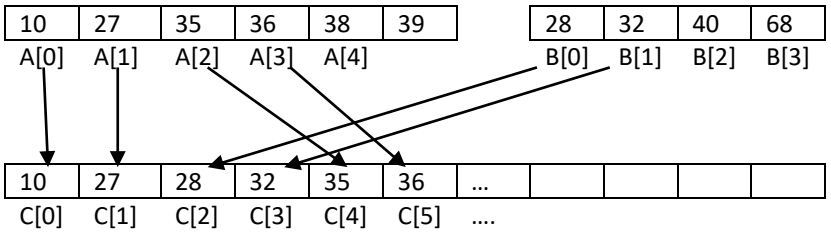
x^0	x^1	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9
9	2	0	4	0	7	0	0	0	0
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]

ip: enter coeff and degree (if coeff is 0 then stop scanning)

```
 7  5 ↲
4  3 ↲
2  1 ↲
9  0 ↲
0 ↲   stop
// sample code for scanning polynomial equation
int a[10]={0}
while(1)
{       printf("enter coeff and degree (0 to stop) :");
        scanf("%d", &coeff);
        if( coeff==0) break;       // stop scanning
        scanf("%d", &expo);
        a[expo]=coeff;
}
-----
-----
```

Imp 46) Code to merge two sorted array elements into one output array, the output remain in sorted order

ip: A[ ] = {10, 27, 35, 39, 59, 98 }, n1=6; // n1 is no.of elements in the array A[]
B[ ] = {28, 32, 40, 68 }, n2=4; // n2 is no.of elements in the array B[]
op: C[ ] = {10, 27, 28, 32, 35, 39, 40, 59, 68, 98 }



step1: let array A[] contained n1 elements with sorted order (input in sorted order)
step2: let array B[] contained n2 elements with sorted order (input in sorted order)
step3: take index variables with i=0, j=0, k=0
step4: while(i<n1 && j<n2)
if(A[i] < B[j]) C[k++]=A[i++];
else C[k++]=B[j++];
after above loop, some elements remain in A[] or B[], thereafter copy them to C[], this is as given below
while(i<n1) // after above loop, still if i < n1 means some elements left in A[], so copy to C[]
C[k++]=A[i++];
while(j<n2) // after above loop, still if j < n2 means some elements left in B[], so copy to C[]
C[k++]=B[j++];
step5: now C[] contained k elements (n1+n2), print all

---

47) following program simulate the bus reservation process, here repeatedly tickets are issued one by one to passengers, at end of process, the seat number -1 is entered as end of reservation. Finally print list of reserved seats.

step1: take array: int a[40]={0}; // let 40 is the maximum number of seats
step2: Initialize all cells with 0 values, as it indicates all seats are free at the beginning;
(0 indicates unreserved, 1 indicates reserved)
step3: scan the seat number to 'x';
step4: if(x == -1) stop the program(stop reservation), go to step6;
step5: if a[x]==1 then display error message "seat already reserved"
else a[x]=1; // reserving seat numbers: 0→un-reserved, 1→reserved
goto step3;
step6: print all seats which are reserved
step7: stop
input:
enter seat no: 1
enter seat no: 3
enter seat no: 1 (error, already reserved)
enter seat no: 7
enter seat no: -1 (end of program)
output: reserved seats are: 1, 3, 7

---

48) In a school, for 2nd standard children, a mathematical test is conducting through online, where addition of two values should be calculated. The values are single-digit numbers and they have to face 10 questions. The output is, how many questions correctly answered by the children. Let us see sample input/output

```
Question 1: 1+2
your answer: 3 ↵

Question 2: 2+8
your answer: 9 ↵

Question 3:  2+3
your answer:  5 ↵
```

output is: Answered correctly 2 out of 3 questions.

the sample code is:

```
void main()
{    int i,k1,k2,ans, count=0;
     for(i=0; i<10; i++)
     {       k1=rand()%9+1;
             k2=rand()%9+1;
             printf("Question %d:  %d + %d", i, k1,k2);
             printf("\n your answer : ");
             scanf("%d", &ans);
             if(ans==k1+k2) count++;
     }
     -------
```

Logic: the library function **rand()** generates the random numbers in between 0-65535.

for example, k=rand(); // the k value can be 0 to 65535

but to generate 2-digit number use k=rand()%9+1; → this generates a number in between 1-9.

hint: use header file 'stdlib.h' for rand() function

note: rand() function may generate duplicate values, the same numbers may generate again and again. that is, in the above code, the same question may repeated more than once. So solve yourself by keeping track of previous question values in the array for duplicate checking for the next questions.

---

49) Code to exchange all odd number to beginning of array and even numbers to end of array

Process: take index variables i , j and set i=0, j=n-1, now select even number in left side of array using 'i' and select odd numbers in right side using 'j' (opposite direction) and exchange for every pair of selection.

ip: a[ ] = { 10, 21, 30, 37, 40, 87, 44, 11 };

op: a[ ] = { 11, 21, 87, 37, 40, 44, 30, 10 } // after exchange

the logic for exchange elements is as follows

```
while(1)
{   while( i<j && a[i]%2==1)
        i++;    // if odd then skip by i++, if even then select by stopping loop
    while( i<j && a[j]%2==0)
        j--;    // if even then skip by j--, if odd then select by stopping loop.
    if( i<j )
        swap( &a[i], &a[j]);  // exchange for every pair of selection
    else break; // stop when all are exchanged.
}
```

---

50) Code to accept 5 values one by one from keyboard, while scanning values, the next input should not be less than the previous value, if user entered by mistake then reject it. Finally print all values and also observe whether in ascending order not?

Observe following ip/op

input: enter value 1: 12
enter value 2: 16
enter value 3: 7
this input value '7' rejected (because it is less than previous value)
enter value 3: 19
enter value 4: 23
enter value 5: 31

output: 12 16 19 23 31

```
previous=0;     // first value index
for(i=0;  i<5;  )
{       scan( a[i] );
        if( a[i] < a[previous] )        // for first time, a[0] compared with a[0] only
            print("this input value %d  rejected", a[i] );
        else
            previous=i++;       // now previous value index is 'i' for next coming element.
                                //  i++ means , the a[i] accepted as input, so go to next value.
}
-----
```

---

51) Code to accept 5 values one by one from KB, do not allow duplicate values while scanning.
Finally print all values.

input: enter value 1: 12
enter value 2: 16
enter value 3: 12
the input value '12' is duplicate (already entered), rejected
enter value 3: 19
enter value 4: 23
enter value 5: 30

output: 12 16 19 23 30

```
for(i=0;  i<5;  )
{       scan( k );
        bool=false;
        for(j=0; j<i; j++)    // loop for comparing k with previous scanned values
                if a[j] == k
                        bool=true and stop the j-loop
        if( bool==true) print("error, duplicate found\n");
        else  a[i++]=k;
}
print 5 values here
```

---

52) Code to accept 5 values one by one from keyboard, while scanning values, automatically arrange in ascending order (do not use any sorting technique)

logic: After scanning a value, compare with previous elements in the array, if they are bigger, then shift all bigger elements to right side and then insert scanned value in that gap.

ip: enter value 1: 12 → a[]=[**12**]
enter value 2: 19 → a[]=[12,**19**]
enter value 3: 5 → a[]=[**5**,10,19]
// here shift 10,19 to next positions(right side), so that we get a gap at A[0], where insert 5
enter value 4: 24 → a[]=[10,12,19,**24**]
enter value 5: 15 → a[]=[10,12,**15**,19,24]
// here shift 19,24 to next positions(right side), so that we get a gap at A[2], where insert 15

```
for(i=0;  i<5; i++ )
{       scan( k );
        for(j=i-1; j>=0; j-- )      // loop for comparing a[i] with previous positioned values
                if (a[j]>k)
                    a[j+1]=a[j];    // if a[j] is bigger than k, then shift to right side
                else break;         // if small found then stop the shifting
        a[j+1]=k;                   // inserting 'k' in right position
}
here print 5 values.
```

Functions

1) some model code already given in this examples, fill the uncompleted code.(don't change existing code)

```
void  main()
{       int a=10, b=20;
        c=adder( 11 , 22 );                 // calling with arguments 11 , 22
        printf("\n sum=%d", c);
        c=adder( 111 , 222 );               // calling with arguments 111 , 222
        printf("\n sum=%d", c);
        c=adder( a , b );                   // calling with arguments 10 , 20  (values of a , b)
        printf("\n sum=%d", c);
}
int  adder( int x , int y )
{       ----                                // fill this block with your code
        ----
}
```

---

2) Write a function(fn) called **big(),** which takes two integer arguments and returns the big of arguments, later write a main() fn where scan 3 values and find big of them.

```
void main()
{       int a, b, c, k;
        scan  a, b, c;
        k=big(a,b);
        k=big(k,c);        //  we can also call like: k=big( big(a,b), c )  → embedded calling
        print  k;
}
int big( int x, int y )
{       --------
        --------
}
```

---

3) Write a fn called **findTax(),** which takes salary as argument and returns the tax.

tax calculation is: if(salary<=10000) tax is zero.
if(salary>10000 and salary<=20000) tax is 5% on salary.
if(salary>20000) tax is 8% on salary.

```
void main()
{       float salary, tax;
        printf("enter salary :");
        scanf("%f", &salary);
        tax=findTax(salary);            // function call
        printf("tax  is  %f", tax);
}
float findTax(float  salary)
{       -------
        -------
}
```

---

4) Write a fn to calculate x^y value, here the fn takes x,y as arguments and returns the x^y value.
Later write a main() fn to find 2^3 and 3^2.

```
void main()
{       int k;
        k=power( 2 , 3 );          // call-1 for 2^3
        printf("2^3 = %d", k);
        k=power( 3 , 2 );          // call-2 for 3^2
        printf("3^2 = %d", k);
}
int power( int x ,  int y )
{   -------
}
```

---

5) Write a fn to find factorial (eg: 4! → 4*3*2*1) of a given number, later find n_{Cr} in main() fn.
n_{Cr} → n! / ((n-r)! * r!) the main() fn is as given below

```
void main()
{       int n, r;                  // to store input values
        int f1, f2, f3;            // f1 to store n!,  f2 to store r!, f3 to store n-r!
        printf("enter n , r values :");
        scanf("%d%d", &n, &r);
        f1=fact(n);                // call-1 for n!
        f2=fact(r);                // call-2 for r!
        f3=fact(n-r);              // call-3 for n-r!
        f1=f1/(f2*f3);
        printf("\n ncr=%d", f1);
}
int fact( int n )
{       ------
        ------
}
```

---

6) Write a fn to find sum of 1+2+3+ ……+N (without using formula), this function takes **'N'** as argument and returns the sum of 1 to **N**. Later write a main() function and check this function return value is equal to N(N+1)/2 or not?

```
void main()
{       int N, result;
        result=findSum(N);     // function  call
        if(result==N*(N+1)/2)
              printf("equal");
        else printf("not equal");
}
int findSum( int N )
{       -------
        -------
}
```

---

7) Write a fn called **sumOfCubes(),** which returns the sum of cubes of all digits of a given integer argument , later write main() fn where scan N as input from KB and print it is Armstrong or not?

ip: 135 ip: 153 ($1^3+5^3+3^3$→1+125+27→153)

op: not a Armstrong op: Armstrong

logic to get sum of cubes of all digits is

```
while(n>0)
{   r = n%10;
    sum=sum+r*r*r;
    n=n/10;
}
```

---

8) Write a fn called **reverse(),** which returns the reverse of a given integer argument, later write a main() fn where scan **N** as input from KB and print whether it is palindrome or not?

ip: 2345 ip: 232

op: not a palindrome op: yes palindrome

logic to get reverse of N is

```
while(n>0)
{   r = n%10;
    rev=rev*10+r;
    n=n/10;
}
```

---

9) Write a fn called **sumOfDivisors(),** which returns the sum of all divisors of a given int argument N .

L**ogic:** Check with all possible divisors from 1 to N/2 and add all divisible to variable 'sum' later return it.

the fn proto type is: **int sumOfDivisors(int);**

Later write a main() fn and check the input value N is perfect or not?

perfect: if sum of all divisors is equal to given N then it is said to be perfect. (Exclude N as divisor)

For example: 6 (1+2+3→6), 28(1+2+4+7+14→28)

ip: enter N value: 6 ip: enter N value: 7 ip: enter N value: 28

op: yes op: no op: yes

---

10) Code a function called **"next9Divisible()",** this takes integer N argument and returns next divisible of 9. If input is 14 then next 9 divisible is 18, if input is 26 then next divisible is 27, if input 27 then next is 36.

```
… next9Divisible( …)
{    ----
}
void main()
{    ----
}
```

---

11) Write a fn "**isPrime()",** which returns the boolean value for given argument N.

If given argument **N** is prime then this function returns 1(true) or else returns 0(false).

Later write a main() fn where take a loop and print list of primes from 50 to 100.

fn proto-type is: **int isPrime(int);**

---

12) Write a fn "**isPrime()",** which returns the boolean value for given argument N.
print list of twin primes between 1 to 100, for example: 3-5, 5-7, 11-13, 17-19, etc.

---

13) Write a fn called **sumOfCubes(),** which returns the sum of cubes of all digits of a given 'int' argument**.** Using **sumOfCubes()** fn, write a main() fn and print list of Armstrong numbers which are in b/w **1 to 1000.**
output: 1 , 153 , 370 , 371 , 407

---

14) Let in main() function there is an array A[] with 10 predefined values, now yourself write **isPrime() f**unction's body and its call statement to count primes in the array. The function takes 'int' as argument and returns Boolean value.

```
void main()
{       int a[10]={10, 18, 11, 15, 17, 21, 23, 5, 30, 31};
        int count=0, i;
        for(i=0; i<10; i++)
        {       -------
        }
        printf("\n count = %d", count);                  // count = 5
}
int isPrime( int n )
{       ---------
}
```

---

15) Let at main() function there is an array A[] with 10 predefined values, now yourself write **reverse()** function's body and its call statement to count palindromes in the array. The function takes integer N as argument and returns the reverse of N value.

```
void main()
{       int a[10]={10, 181, 101, 15, 1771, 12321, 123, 5555, 30, 31};
        int count=0,i;
        ------
        -------
}
```

---

16) Write a fn called **printTable()** which prints multiplication table upto 10 terms. (this fn returns nothing) this function takes integer argument N as table number and prints multiplication table for 10 terms.
If input is 8 then output is: 8*1=8, 8*2=16, 8*3=24,….., 8*10=80.
In the main() fn, call printTable() for 2 times to print 8 and 13 tables.

```
void main()
{       printTable(8);
        -----
}
void printTable(int n)
{       --------
}
```

---

17) using above **printTable() fn**, write a main() fn and print all multiplication tables from 1 to 20.

---

18) Write a fn called **printAllTables()** which prints tables in given limits. (this fn returns nothing)
this function takes two integer arguments as (lower, upper) and prints all tables between them.
In the main() fn, if called like "printAllTables(3, 8)" then it prints tables from 3 to 8.

```
void printAllTables(int low, int upper)
{       --------
}
```

---

19) Write a fn called **printAllDivisibles(),** which prints all divisibles of a given integer argument .
Later write a main() fn, where repeatedly scan N value until N==0, and print all divisibles of each input of N.

```
ip: 15↵
op: 1, 3, 5, 15
ip: 8↵
op: 1, 2, 4, 8
ip: 9↵
op: 1, 3, 9
ip: 0↵  ( program stops, when N==0 )
```

```
void main()
{       int N;
        while(1)
        {       printf("enter N value (at end type 0):");
                scanf("%d", &N);
                if(N==0) break;
                -----
        }
}
void printAllDivisibles( int n )
{   …
}
```

---

20) Write a function called **printDigit()**, which prints given digit in English words. For example, if input argument is 4 then prints "four" as English word. Now using this fn, write a main() fn, where scan 3-digit number like 247 and print output as **two-four-seven.**

```
void main()
{       int n=247;
        printDigit(n/100);
        printDigit(n/10%10);
        printDigit(n%10);
}
void printDigit(int n)
{       ----------
}
```

---

21) Write a fn called **sumOfSquares(),** which returns sum of squares of 1 to 10. ($1^2 + 2^2 + 3^2 + 4^2 ... + 10^2$)

the fn proto type is: **int sumOfSquares();**

this fn takes no arguments but returns sum of squares of 1 to 10 numbers.

Later write a main() fn and check this sum is equal to 385 or not?. (output is: yes/no)

```
void main()
{    ----------
}
int sumOfSquares()        // remember, no parameters required here (bcoz need to find first 10 terms)
{    ----------
}
```

---

22) Demo: void main()

```
{       int N;
        N=scanValue();
        if( N%2==0)   printf("even");
        else   printf("odd");
}
int scanValue()
{       int k;
        printf("enter  a  value::");
        scanf("%d", &k);
        return k;
}
```

The function scanValue() reads a value from keyboard and returns to main() fn, this value is receiving by N and printing whether it is odd/even.

---

23) Write a fn called readMarks(), which scans marks of one subject at a time and returns it.
This input marks must be in between 0-100, if not then scan again and again until user entered a valid input 0-100, later return this marks. Now write main() fn, where scan two subject marks and print pass/fail.
If he got >=50 in 2 subjects then print "pass" or else "fail".

```
void main()
{       int m1,m2;
        m1=readMarks();
        m2=readMarks();
        ------
}
int readMarks()
{       int M;
        while(1)
        {       // here scan 'M' value, if valid then stop the loop and return M
                // if not then scan again until valid.
        }
        return M;
}
```

---

24) Write 2 functions called **fact()** and **n_{Cr}()**; the function **n_{Cr}()** takes the help of fact() while calculating factorial values. Later write a main() function to check nCr() is working properly or not? Eg: $7c_3 \rightarrow 35$

```
void main()
{       // here call nCr() function to print nCr value
}
int nCr(int n, int r)
{
        //  here call fact() function for 3 times to calculate n!, r! and n-r! values.
}
int fact(int n)
{
        // find factorial value here
}
```

---

25) Write a fn called **getSumOrProduct()**, which returns sum/product of 1 to N. This function takes N & flag as arguments. if flag==0 then function returns 1+2+3+….+N. if flag==1 then fn returns 1*2*3* ….*N.
The main() fn already given

```
int getSumOrProduct(int N, int flag)
{       ----------
        ----------
}
void main()
{       int N=5, sum, product ;
        sum=getSumOrProduct(N, 0 );             // returns 1+2+3+…+N because flag=0
        product=getSumOrProduct(N, 1);          // returns 1*2*3*…*N because flag=1
        printf("\n sum is %d,  product is %d", sum, product);
}
```

note: we may get one doubt, why can't we return two values at a time?, because, the C function designed to return only one value at a time using return statement, so we can't return two or more values at a time, but using pointers we can return more values, that we will see in next chapter.

Home Work

---

30) Suppose we have a fn called **big3(),** which takes three arguments and returns a big value, but we need to find big of 2 values using such big3() fn, now explore how to call and find big of 2 values.

note: Here in the main fn, we scan only 2 values as input to find big of them (little tricky question)

```
void main()
{       int a, b, result;
        printf("enter  any 2 values :");
        scan("%d%d", &a, &b);
        ---------
}
int big3( int x, int y, int z )
{       --------
        --------
}
```

---

31) Write a function which takes 3 arguments as date, and returns valid or not? Later write a main() fn where scan the date, if not valid then scan again and again until user entered a valid date.

Finally print such valid date.

ip: 31-2-2001
op: invalid date, try again
ip: 21-2-2001
op: yes, valid date (stop the program)

---

32) Write a main() fn, where scan 5 values to array a[], and print each number multiplies using function.

```
void main()
{       int a[5]={10, 18, 11, 15, 17};
        int count=0, i;
        for(i=0; i<5; i++)
        {   -----------
        }
}
void printMultiples(int n)
{   -----------
}
```

10→1,2,5,10
18→1,2,6,9,18 …

---

33) Write a fn called digitSum(), this returns sum of all digits of a given number, later write a main fn where scan N value, and repeatedly sum of digits until single digit reached. 19999=>1+9+9+9+9=>37 37=>3+7=>10 10=>1+0=>1 Here: here repeatedly call the digitSum() until sum of digits become < 10 (single digit)

---

34) Write a program to scan values one by one until last input is zero, when zero is entered then stop scanning. Here insert each scanned value into array if it is a prime. If not prime then skip such value. use **isPrime()** function which we used before examples. Finally print all values of array(all primes).

ip: 12 17 18 20 31 22 13 11 10 47 0 (0 to stop).
op: 17 31 13 11 47 (all are primes)

```
void main()
{   int a[20], count=0, N,  …. etc.
    while(1)
    {       scan N value here
            if N==0  then stop the loop
            here check N is prime or not using function.
            if N is prime then
            {       a[count]=N;     // inserting N into array
                    count++;
            }
    }
    here print all primes of a[i] , where i = 0 to count
}
```

---

35) Write a fn "**nextPrime()",** which returns the next prime of given argument N. If N is 11 then it returns 13 if N is 14 then it returns 17, If N is 17 then it returns 19.
Later write a main() fn where print list of primes from 10 to 1000.
fn proto-type is: **int nextPrime(int);**

---

36) Write two functions 1) binary to decimal 2) decimal to binary [main() function already given]
Function1) takes binary number as argument and returns decimal number: **int binaryToDecimal(int);**
Function2) takes decimal number as argument and returns binary number: **int decimalToBinary(int);**

```
void  main()
{   printf("%d ",   binaryToDecimal(1101) );
    printf("%d ",   decimalToBinary(13) );
}
```

---

37) Write a fn called **gcd()**, which takes 2 numbers as arguments and returns greatest common factor. Write main() fn, where scan N values to array a[], and find their gcd. For example

ip : arr[] = {12, 40, 60, 80, 100}
op : 4
ip : arr[] = {1, 2, 3}
op : 1

Recursion

Guess the output of following programs

1)
```
void main()
{       show( 10 );
}
void show( int  N )
{       if( N==0 )
            return;
        show( N-1 );
        printf(" %d  ", N );
}
```

---

2)
```
void main()
{       show( 100 );
}
void show( int  N )
{       if( N==0 )
            return;
        show( N/2 );
        printf ( "%d  ", N );
}
```

---

3)
```
void main()
{       show( 1 );
}
void show( int  p )
{       if( p>100 )
            return;
        printf("%d  ", p );
        show( p*2 );
}
```

---

4)
```
void main()
{       show(1);
}
void show( int  N )
{       if(N == 6)
        {   printf("  and  ");
            return;
        }
        printf("  hello  ");
        show( N + 1 );
        printf("  world  ");
}
```

---

5)
```
void main()
{       show( 1 , 5 );
}
void show( int  p , int  count )
{       if( count==0 )
            return;
        printf ( "%d  ",  p  );  ?
        show( p*2 ,  count-1 );
}
```

---

6) Guess the output
```
void main()
{       int N=8;
        show( N, 10 );
}
void show( int N, int i )
{       if( i==0) return;
        show ( N , i-1 );
        printf("\n %d * %d = %d", N, i, N*i );
}
```

---

7) Guess the output
```
void main()
{       int N=8;
        show(0,  1,  N);
}
void show( int x, int y, int N )
{       if(N==0) return;
        print("%d ",  x );
        show( y,  x+y, N-1);
}
```

---

8) Decimal to Binary output
```
void main()
{       show( 13 );
}
void  show( int  N  )
{       if( N==0 )
            return;
        show( N/2 );
        printf("%d", N%2 );
}
```

---

9) Program to add 8+8+8 … 4 times (it doesn't work, check out)

```
void main()
{       k=find( 4 );
        printf(" %d  " ,  k );    ?
}
int  find ( int  N )
{       int sum=0 , k;
        if( N==0 )  return(sum);
        sum=sum+8;
        k=find( N-1 );
        return k;
}
```

---

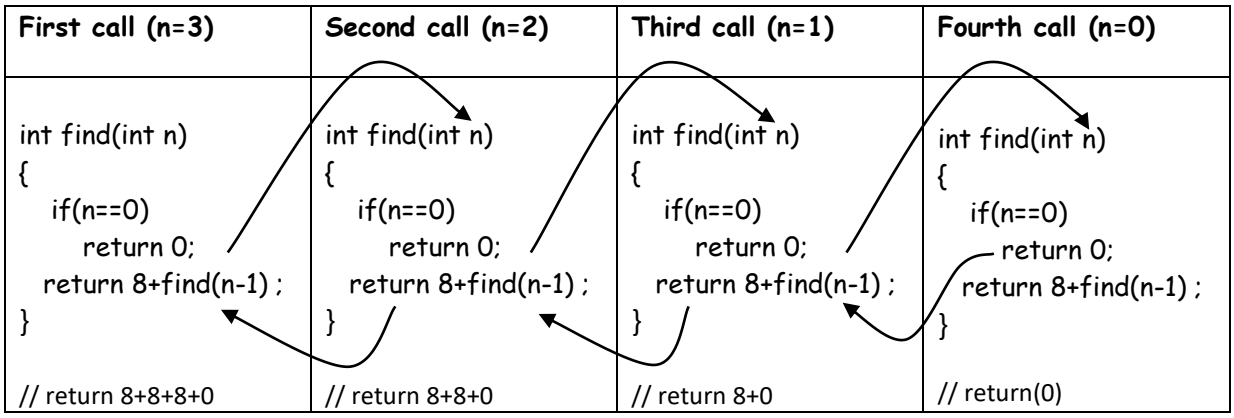
10) small modification of above program (guess the output)

```
void main()
{     k = find( 4 , 0 );
      printf( " %d  ", k );    ?
}
int find ( int  N  ,  int  sum )
{       int  k;
        if( N==0 ) return(sum);
        sum=sum+8;
        k = find( N-1 ,  sum );
        return k;
}
```

---

11) small modification of above program (simplified)

```
void main()
{       k=find( 4 );
        printf( " %d  ", k );    ?
}
int  find ( int  N  )
{       if( N==0 )  return 0;
        return  8 + find( N-1);
}
```



---

12) Program to count no.of digits in N (guess the output)

```
void main()
{       int N=3456 ,  k ;
        k=find( N  );
        printf("%d  ",  k );    ?
}
int  find ( int  N )
{       int count=0;
        if( N==0 )
            return count;
        count++;
        return find( N/10 );
}
```

---

13) small modification of above program (guess the output)

```
void main()
{       k=find( 3456 , 0 );
        printf("%d  ",  k );    ?
}
int  find ( int  N , int count )
{       if( N==0 )
            return count;
        return  find( N/10 , count+1 );
}
```

---

14) small modification of above program (guess the output)

```
void main()
{       k=find( 3456 );
        printf("%d  ",  k );
}
int  find ( int  N )
{       if( N==0 )
            return 0 ;
        return  1+find( N/10 );
}
```

---

15) Guess the output

```
void main()
{       k=find( 345 );
        printf( " %d  ", k );     ?
}
int  find ( int  N  )
{       if( N==0 )  return 0;
        return  N%10 + find( N/10 );
}
```

Home Work

Write solution for the following programs

(don't use any loop control structure in the following examples)

don't change existing code, only fill the dotted lines

17) Fill the body of recursive function **show()** to print 1 to 10 numbers.

```
void main()
{       show(1);
}
void show( int  i )
{       -----
        -----
}
```

---

18) Write a recursive function to print string "hello" for N times

op: hello hello hello …..N times

```
void main()
{       int N;
        scanf("%d", &N);
        show(N);
}
void show( int  N )
{       -----
        -----
}
```

---

19) Fill the body of recursive function **show()** to print 1 to N numbers, this function takes N as argument and prints the numbers from 1 to N.

ip: enter N value: 13 ↵

op: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13.

```
void main()
{       int N;
        scanf("%d", &N);
        show(N);
}
void show( int  N )
{       -----
        -----
}
```

---

20) write a recursive function to print following output N, N/2, N/4, N/8, N/16,... 1

This function takes N as argument and prints output from N to 1

ip: N is 100

op: 100, 50, 25, 12, 6, 3, 1

---

21) Generate and print list of numbers from N to 1, here N is input from keyboard and print list of numbers as long as the value of N becomes 1.

if N is even then next number of N is → N/2 (half)

if N is odd then next number of N is → 3N + 1

if input N is 13, then we have to print as: 13, 40, 20, 10, 5, 16, 8, 4, 2, 1

```
void main()
{       int N=13;                       // or else scan N value from KB
        show(N);
}
void  show(int N)
{       ------
        -----
}
```

---

22) Write a recursive function to print following output 1, 2, 4, 8, 16, 32, 64 up to N terms.

```
void main()
{       int count=10;
        show(1, count);       // 'count' is number of terms to print, where '1' is starting value of series.
}
void show(int P, int count )  // 'count' downs from 10 to 0, where 'P' raises its value to 1, 2, 4, 8, ….
{       -------
        -------
}
```

---

23) Write a recursive function(s) to print following output (don't use any loop)

```
12345
12345
------
8 rows
```

```
void main()
{       int N=8;                  // no.of rows to print
        show(N);
}
void show(int N)
{       ---------
        printRow ( 5 );           //  this function prints 1 to 5 values of a row
        ---------                 //  call show() recursively 8 times, for 8 rows
}
void printRow ( int i )
{       if(i==0) return;
        printRow( i-1 );
        printf("%d ", i );        // prints each row here
}
```

---

24) Write a recursive function(s) to print following output (don't use any loop)

```
12345678
1234567
--------
12
1
```

```
void main()
{       int N=8;        // no.of rows to print
        show(N);
}
void show(int N)
{       ------
        printRow( --- );  // this function prints 1 to N values of row
        ------
}
void printRow(int N)
{       -----           // prints each row here
        -----
}
```

---

25) Write a recursive function to find sum of 2+2+2+2+2+ ….+N times, here N is input value.

note: do not use multiplication operator(*) in the program

ip: enter N value:6

op: output = 12

```
void main()
{       int k, N=6;
        k=find(N);
        printf("sums of 2 for N times is %d", k);
}
int find(int N)
{       ----
}
```

---

26) Write recursive function to find sum of N. (1+2+3+ …. +N)

ip: N=5

op: 5+4+3+2+1

```
void main()
{       int k;
        k=find(5);
        printf("sum of 1 to N is %d", k);
}
int find( int N )
{       ----
        ----
}
```

---

27) The sum of squares of first ten natural numbers is $1^2 + 2^2 + 3^2 ... + 10^2 = 385$

Write a program to prove it (output is “yes” or “no”)

```
void main()
{       int k;
        k=find(10);
        if( k==385 ) printf(“yes, program is right”);
        else  printf(“no, program has some bugs”);
}
int find( int N )
{       ----            // stop when N becomes 0.
        ----
}
```

Note: Generally, mistakes showing by compiler is said to be syntax-errors, whereas logical errors are called bugs. Logical error means, here the program successfully compiled but shows wrong output.

---

28) Write recursive function to find $\mathbf{X^Y}$, where X is base and Y is exponent. the recurrence relation is

```
f(X,Y) → X*f(X,Y-1)    if Y>0
f(X,Y) → 1             if Y==0
void  main()
{     int X=3, Y=4, Z;
      Z=power( X,Y);
      printf(“ X^Y = %d”, Z);
}
int power(int X, int Y)
{       --------
        --------
}
```

---

29) Write a recursive function to find factorial of N. (1*2*3* …. *N)

the recurrence relation is:

```
f(n) → n*f(n-1)   if n>0
f(n) → 1          if n==1 or 0
```

for example

```
ip: N=5
op: 5*4*3*2*1 →120
void main()
{       int k;
        k=factorial(5);
        printf(“factorial of 5 is %d”, k);
}
int factorial(int N)
{       ----
        ----
}
```

---

30) Write a recursive functions to print list of factorial values 1, 2, 6, 24, 120, 720 up to 8 values

op: 1, 2, 6, 24, 120, 720 (1, 1*2, 1*2*3, 1*2*3*4, …)

```
void main()
{       int N=8;
        printAll( N );
}
void printAll( int N )
{       int  k;
        if(N==0)   return;
        ----------                  // recursively call  printAll() here
        k=fact(N);                  // calling factorial function.
        printf("%d ", k);
}
int fact(int N)                     // this is also recursive function
{       ----
        ----
}
```

---

31) Write a recursive function to print sum of 1+2+4+8+16...N times. ($2^0+2^1+2^2+2^3+$ …. N times)

Hint: do not use pow() library function.

ip: N=5

op: 31

```
void main()
{       int k, N=5, sum;
        sum=find(1, N);     // initial value of series is 1, and 'N' is number of terms
        printf("sum of powers is %d", sum);
}
int  find(int p , int N)
{       ------
        ------
}
```

---

32) Write a recursive function to return smallest factor of given N.

If N=15 then return 3, if N=77 then return 7, if N is 17 then return 17.

---

33) Write a recursive function to return biggest factor of given N. (other than N)

If N=15 then return 5, if N=77 then return 11, if N is 17 then return 1.

---

34) write a recursive function to print last odd digit of a given number, if odd not found then return -1.

ip: 23458	ip: 3456	ip: 486
op: 5	op: 5	op: -1

---

35) Write recursive function to find sum of factors of given N. (exclude N as factor)
Later write a main() function and check given number is perfect or not?,
if N is perfect then sum of factors is equal to N. (Check for factors from 1 to N/2)

input: 6 input:28 input: 10
output: yes, perfect output: yes, perfect output: no, not perfect

```
void main()
{       int k,N;
        scanf("%d", &N);
        k=sumFactors(N, 1);          // intial value of 'i' is 1
        if( N==k)  printf("yes, it is perfect");
        else  printf("no, it is not perfect");
}
int sumFactors( int N, int i )
{       ------                       // stop the process when i>N/2
}
```

---

36) Write a recursive function checkPrime(), to check given number is prime or not?
This function returns (1/0 →bool value)

ip: 13 ip: 15
op: yes, it is prime op: no, it is not prime

```
void main()
{       int N=13, bool;
        bool=checkPrime(2, N);      // check prime ness by dividing from 2 to N/2
        if(bool==1) printf("N is prime");
        else  printf("N is not prime");
}
int checkPrime(int i, int N)
{       ----
}
```

---

37) Find biggest digit in a given number using recursion

```
void main()
{       int n=5173;
        k=findBig(n , 0);
        printf("biggest is %d", k );
}
int findBig( int n, int big )      // the parameter 'big' holds the biggest digit of N, initial value is zero.
{       ------
        ------
}
```

---

38) justify which code is best in following recursive functions, which finds sum of 1+2+3+…..+N

```
Void main()
{   int n=4, sum;
    sum=find(n, 0);
    printf("sum=%d", sum);
}

int find(int n, int s)
{    if(n==0)
          return s;
     s=s+n;
     return find(n-1,s);
}
```

```
void main()
{   int n=4, sum;
    sum=find(n);
    printf("sum=%d", sum);
}

int find(int n)
{   if(n==0)
          return 0;
    return n+find(n-1);
}
```

---

39) Write a recursive function printFibo(), to print fibonacci numbers in between 10 & 200

op: ~~0, 1, 1, 2, 3, 5, 8,~~ 13, 21, 34, 55, 84, 139, ~~223, 362, 585,~~

this main() fn is as follows

```
void main()
{      printFibo( 0, 1, 10, 200);          // 0,1 are first two numbers in Fibonacci series, 10-200 are limits
}
void printFibo( int x, int y, int lower, int upper)
{         ----------
}
```

---

40) Write a recursive function checkFibo(), to check given number is in Fibonacci series or not?

This function returns (1/0 →bool value)

ip: 13 ip: 15

op: yes, it is in series op: no, not in series

---

41) Write a function to find sum of odd digits only.

ip: 2345

op: odd digits sum = 8 (3+5)

---

42) Write a recursive function to return first digit of given number.

ip: 2345 ip: 456

op: 2 op: 4

logic: repeatedly divide N=N/10 until N>9, finally N contains first digit then return it.

if N>9 means, the N has more than one digit value, so cut it by doing N=N/10

---

43) write a recursive function to print first odd digit of a given number, if odd not found then return -1.

ip: 2345 ip: 4563 ip: 486

op: 3 op: 5 op: -1

---

44) Write a program to find GCD of two numbers. (GCD/HCF → Greatest Common Divisor)

ip: 12, 18

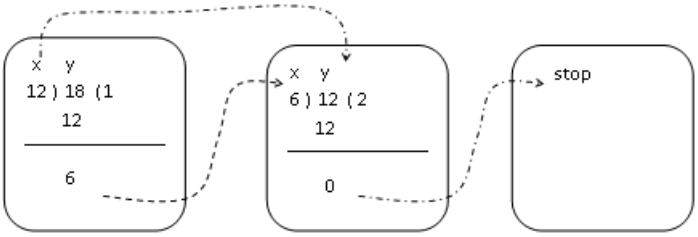
op: 6

1.let x, y are input values

2.divide 'y' with 'x' (irrespective of which is big and which is small)

3.if remainder(R) is zero then stop and print 'x' as GCD

4.if R is not zero then take 'x' as 'y' and 'R' as 'x' and continue this process until R is zero.



---

45) x^1/1! + x^2/2! + x^3/3!..... N times [do not use pow() fn]

ip: x=3, N=5

op: sum= 17.4 [3.0 + 4.5 + 4.5 + 3.375 + 2.025 → 17.4]

```
void main()
{       int k, x=3, N=5, sum;
        sum=find(x, N);     // intial value of series is 1, and N is number of terms
        printf("sum of powers is %d", sum);
}
int find( int x , int N)
{       ------                              // here write stop condition
        return power(x,N)/fact(N) + find(x, N-1);
}
```

// here power() and fact() are again recursive function (write yourself)

---

46) Write a program to find LCM of 3 numbers

ip: 20 15 35

op: 420 (2*2*3*5*7)

1. Let x, y, z are three input numbers
2. Start finding LCM of three with first factor 2 (i=2)
3. Now divide each x, y, z with 2. If any x, y, z divided then take 2 as one multiple in LCM. (LCM=LCM*2) also decrement divisible numbers to quotient obtained in the division
5. Now repeat step-3, step-4 as long as 2 divide any of x, y, z
6. if 2 no longer divided, then next try with next factors 3, after 4, 5…etc, (i++)

repeat this process until any of these (x, y, z) > 1. for example, while(x>1 || y>1 || z>1) {….}

Let the numbers are 20, 15, 35 and following table shows how …

2	20	15	35
2	10	15	35
~~2,~~ 3	5	15	35
~~3,4,~~5	5	5	35
~~5,6,~~7	1	1	7
	1	1	1

---

47) Guess the output of following program

```
void main()
{       int a[5]={ 10 , 20 , 30 , 40 , 50 } ;
        show( a , 5 );
}
void show( int *p ,  int  n )
{       if( n==0)  return;
        printf("%d  ",  *p );
        show( p+1 , n-1 );
}
```

---

48) Guess the output of following program

```
void main()
{       int a[5]={ 10 , 20 , 30 , 40 , 50 } ;
        show( a , 5 );
}
void show( int *p ,  int  n )
{       if( n==0)  return;
         show( p+1 , n-1 );
        printf("%d  ",  *p );
}
```

---

49) Guess the output of following program

```
void main()
{       show( "Hello" );
}
void  show ( char *p  )
{       if( *p=='\0' )
            return;
        show( p+1 );
        printf("%c",  *p );
}
```

---

50) Guess the output of following program

```
void main()
{       show( "APPLE" );
}
void show( char *p  )
{       if( *p=='\0' )
            return;
        printf("%s\n",  p );
        show( p+1 );
}
```

---

51) Write a recursive function to print following output

ip: APPLE

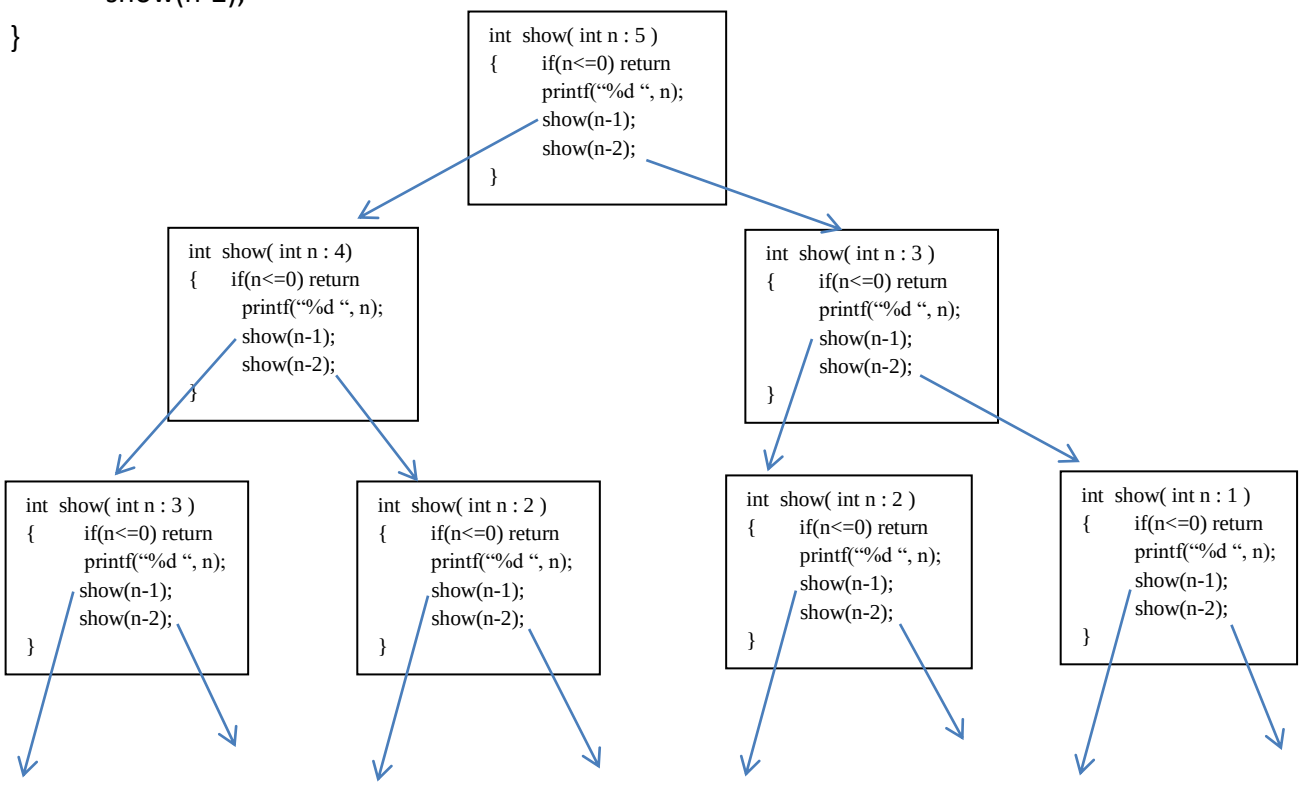
op: E
LE
PLE
PPLE
APPLE

```
void main()
{       show("APPLE");
}
void show(char *p)
{       ------
}
```

---

52) Guess the output of following program **(double calling recursion)**

```
void main()
{       int n=5;
        show(n);
}
void show( int n )
{       if (n<=0) return;
        prinf("%d  ",  n );
        show(n-1);
        show(n-2);
}
```



53) Write a recursive function to print 2-D array of size 3x4 data

```
void main()
{       int a[3][4]= { {4,3,6,1}, {8,7,4,1}, {10,3,2,1} };
        show1( a, 3, 4 );
}
void show1( int a[][4] , int r ,  int c )
{       ---------                    // call show2() function for r times, pass row  address
}
void show2( int a[] , int c )        // this function prints each row for c times
{       -------
}
```

---

54) Write a program to traverse the entire chess board with Knight (horse). Here the Knight visits the every cell only once and it follows of its movement i.e., it moves only in L shape. This function takes x, y values (coordinates) of first step and displays the order of movements in terms of step number for every cell.
(let board size is 5x5)

					6
		2	7		
				5	
	1	8	3		
					4
			9…		

---

55) Write a recursive function to place 8 ministers in the chessboard of 8*8 size; Here the ministers are arranged so that no minister kills one with another. This function takes first minister position as arguments and remaining ministers are arranged accordingly.

1							
			3				
	2						
		4…					

---

56) Write a program to print all permutations (combinations) of a given string.

```
  Input: abc
Output: abc
        acb
        bac
        bca
        cab
        cba
```

Pointers

In C, functions can't return two or more values using return statement; it can return only one or none. Because syntax provided in that way, so using 'return' statement we can return only one value at most. But using pointers we can return more values indirectly through address, this concept often called returning more values using call-by-reference.

---

1) This example explains call-by-value verses call-by-reference.

```
void main()
{       int a=10, b=20;
        change( a , &b );       // a → call-by-value,  &b → call-by-reference
        printf("\n a is %d ,  b is %d" , a , b );
}
void change( int  x , int  *p )
{       x=111;                  // the value 111 is stored in 'x'  (not in 'a')
        *p=222;                 // the value 222 is stored in 'b'  (not in 'p')
}
```

output: a is 10 , b is 222 (no change in 'a' value, but change in 'b' value)

---

2) This example explains how to return sum & product of two numbers to the main() fn.

```
void main()
{       int a=10,b=20, sum, product;
        find( a , b , &sum , &product );
        printf("\n sum is %d , product is %d", sum, product);
}
void find( int x , int y , int  *ps , int  *pp )
{       *ps=x+y;                        // indirectly stored into 'sum' in main() fn.
        *pp=x*y;                        // indirectly stored into 'product' in main() fn.
}
```

output: sum is 30, product is 200

note: In this way, we can return as many values as we want through pointers, this concept often called call-by-reference. We can't return like **return(sum, product);** this concept is not available in c/c++/java.

---

3) This example explains how the function **find()** returns sum & product of 1 to N to the main() function, This function returns sum (1+2+3+4.....+N), and product(1*2*3*4....*N) indirectly through pointers.

Here '**find()'** fn take arguments 'N' along with **<u>address of variables which receives the returning values</u>**.

```
void main()
{       int N=12, sum, product;
        find( N , &sum , &product );
        printf("\n sum = %d , product = %d ", sum, product );
}
void find( int  N,  int  *x,  int  *y)
{       int i, s=0 ,p=1;        // here i,s,p are local variables, 'i' for looping, 's' for addition,  'p' for product.
        for(i=1; i<=N; i++)
        {       s=s+i;
                p=p*i;
        }
        *x=s;       //  sum=s;      this is called returning values indirectly to main() fn.
        *y=p;       //  product=p;
}
```

---

4) Fill the function body with code, which finds area & perimeter of circle.

```
void main()
{       int radius=5;
        float area, perimeter;
        find(radius, &area, &perimeter);
        printf("\n area = %f", area);
        printf("\n perimeter = %f", perimeter);
}
void find( ----- )
{       ------
        ------
}
```

---

5) Write a fn called **findTax(),** which takes 'salary' and '&tax' as arguments and finds tax as given below.

tax calculation is: if salary<=10000 then tax is zero.
if salary>10000 and salary<=20000 then tax is 5% on salary.
if salary>20000 then tax is 8% on salary.

```
void main()
{   float salary, tax;
    printf("enter salary :");
    scanf("%f", &salary);
    findTax( salary , &tax );      // function call
    printf("tax is %f ", tax);
}
void findTax(....)
{   -------
    -------
}
```

---

6) Write a function called findBigSmall(), which takes a, b, c, &big, &small as arguments and finds big&small of a, b, c and returns them.

```
void main()
{   int  a, b, c, big, small;
    scanf("%d%d%d", &a, &b, &c);
    findBig(a, b, c, &big, &small );
    printf("biggest & small of 3 is %d   %d", big , small );
}
void findBigSmall(….)
{   ----
    ----
}
```

---

7) Write a function called **findBigSmall()**, which takes N as argument and returns the big & small **digit** of N.

ip: N=5824

op: big=8, small=2

```
void main()
{       int N=5824, big, small ;
        findBigSmall( ---------- );
        printf("\n big is  %d , small is  %d ", big , small );
}
void findBigSmall( ---------- )
{       -------
        -------
}
```

Logic to find big digit in N.

```
big=0, small=9;
while(N>0)
{       rem=n%10;
        if(big<rem) big=rem;
        if(small>rem) small=rem;
        n=n/10;
}
```

---

8) Correct the following **incrementBy10()** function and its call statement.

```
void main()
{       int n=25;
        incrementBy10(n);            // correct this code (change call-by-value to call-by-reference )
        printf("n value is  %d ", n); // expected output is 35 not 25.
}
void incrementBy10( int  p )        // correct this function
{       p=p+10;
}
```

---

9) fill the following function body, it should replace a=a+b and b=a-b, but the a-b should be +ve

ip: a=2 , b=7

op: a=9 , b=5

```
void main()
{       int a=2 , b=7;
        replace( --- , --- );
        prinf("a=%d , b=%d", a, b);
}
void replace( --- , --- )
{   ----
    ----
}
```

---

10) Correct the following **swap()** function and its call statement.

```
void main()
{       int a=25, b=35;
        swap( a , b );
        printf(" %d  %d ", a, b);       // expected output is (35,25) not (25,35) .
}
void swap( int p , int q )
{       int t;
        t=p;
        p=q;
        q=t;
}
```

---

11) Complete the body of following **swap2()** function for swapping a , b.
the **swap1()** taking help of **swap2()** for swapping values.

```
void main()
{       int a=25, b=35;
        swap1( &a , &b );
        printf("%d  %d " , a , b);       // expected output is (35,25)  not (25,35).
}
void swap1( int *p, int *q )
{
        swap2( &p , &q );
}
void swap2( ----, ----)
{
        -----
}
```

---

12) Complete the body of following **swap2()** function for swapping a , b.
the **swap1()** taking help of **swap2()** for swapping values. (There is little difference with above example)

```
void main()
{   int a=25, b=35;
    swap1( &a , &b );
    printf("%d  %d ", a, b);              // expected output is (35,25) not (25,35).
}
void swap1( int *p, int *q )
{
    swap2( p, q );
}
void swap2( ----, ----)
{       -----
}
```

---

13) Write 2 functions called **sort()** and **swap()**, used to sort 3 variable values (a, b, c) into ascending order. The sort() fn sorts 3 values into order and takes the help of swap() fn for swapping.

logic for sorting a, b, c is : if(a>b) swap a,b; if(a>c) swap a,c; if(b>c) swap b,c;

ip: 23 12 2 ip: 10 2 5
op: 2 12 23 op: 2 5 10

```
void main()
{       int a, b, c;
        printf("\n Enter 3 values :");
        scanf("%d%d%d", &a, &b, &c);
        sort( ---- );
        printf("\n output is: %d  %d  %d", a, b, c);
}
void sort( ------- )
{       --------
}
void swap(int *p, int *q)
{       --------
}
```

---

14) Write a function to add two times, say the two times are employee working time in 2 shifts, and returns the sum of two times.

ip: 10 40 50 (shift1 time)
07 50 40 (shift2 time)
---
op: 18 31 30 (total working time in 2 shifts)

```
void main()
{       int h1, m1, s1, h2, m2, s2, h, m, s;
        printf("enter time1 :");
        scanf("%d%d%d", &h1, &m1, &s1);
        printf("enter time2 :");
        scanf("%d%d%d", &h2, &m2, &s2);
        add(&h, &m, &s, h1, m1, s1, h2, m2, s2);        // (h,m,s)=(h1,m1,s1) + (h2,m2,s2);
        printf("after adding, output time is : %d : %d : %d", h, m, s);
}

void add(  ---------  )
{       ------
        ------
}

// sample code for adding two times
    N=(h1+h2)*3600 + (m1+m2)*60 + (s1+s2);     // converting total time into seconds
    h=N/3600;
    m=(N%3600)/60;
    s=(N%3600)%60;
```

---

15) Complete the code of **scanTwo() and printTwo()** functions which has to scan and print two values .

```
void main()
{   int a,b;
    scanTwo( ---- );        // follow  call-by-referene
    printTwo( ---- );       // follow  call-by-value
}
void scanTwo( int *p  ,  int *q )
{       printf("enter two values::");
        scanf("%d%d",  p , q );          // here scans two values from keyboard and stores into a,b of main() fn.
        or
        scanf("%d%d",  &*p ,  &*q );
}
void printTwo( --- )
{       -----       // here print  a , b values of main() fn
}
```

---

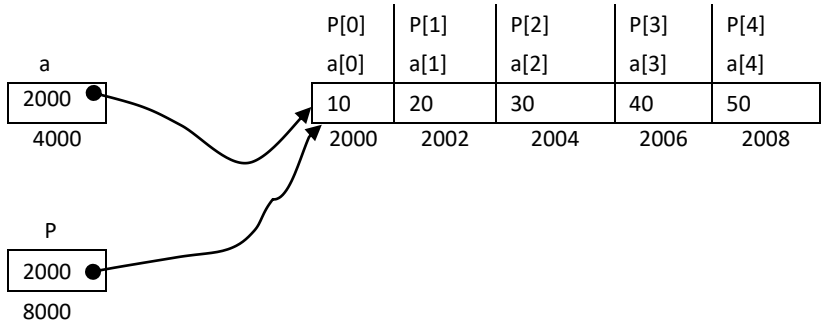
16) Complete the code to find total and average of three numbers using functions.

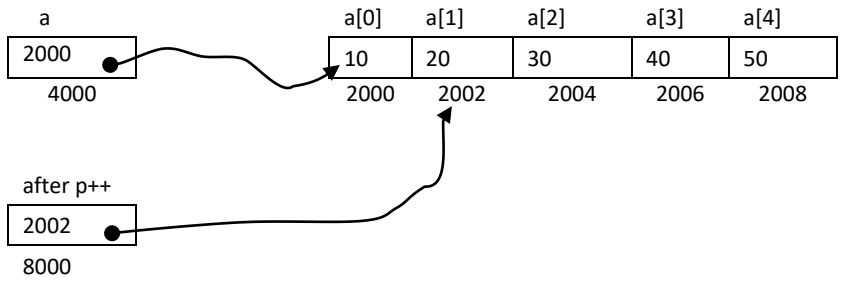
```
void main()
{       int a, b, c, total, average;
        scanThree( …. );
        findTotalAverage( …. );
        printTotalAverage( …. );
}
void   scanThree(…) { …  }
void   findTotalAverage(...) { …. }
void   printTotalAverage(…) { … }
```

Pointer to Array

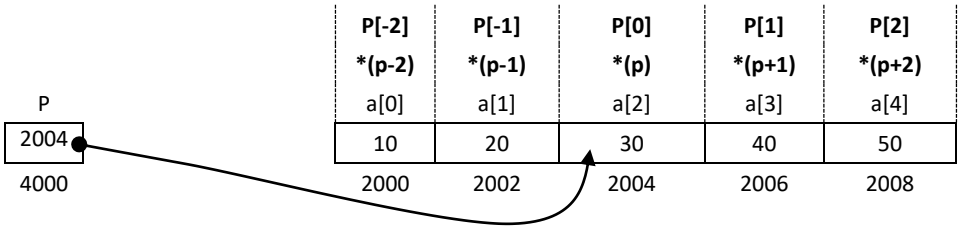
following program explains how to access array elements through pointer

```
void main()
{       int a[5]={10,20,30,40,50};
        int *p;

        p=&a[0];   or   p=a;

        for(i=0; i<5; i++)          //method-1
                printf("%d  ",  *(p+i)  or p[i]  );


        or
        for(i=0; i<5; i++)          //method-2
        {       printf("%d  ",   *p  );
                p++;
        }
}
```





let us see one more example, if the pointer assigned with p=&a[2] , then observe following picture

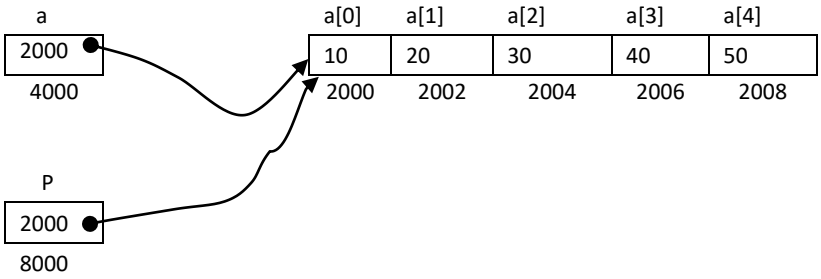


Here p[-2] access a[0], p[-1] access a[1], p[0] access a[2], p[1] access a[3],

```
void main()
{       int a[5] = {10, 20, 30, 40, 50};
        p = &a[2];                  // p=a+2;
        for( i=-2;  i<2;  i++ )
              printf("%d ", p[i]   or  *(p+i) ) ;
}
```

20) Complete the code of function which has to increment each cell value by 1

```
void main()
{   int a[5]={10,20,30,40,50}, i ;
    increment( &a[0] );
    printf("\n array values after incrementing \n ");
    for( i=0; i<5; i++ )
        printf("%d  ", a[i] );     → 11, 21, 31, 41, 51
}

void  increment( int p[ ]  or  int *p )
{       -------
        -------
}
```



note: Remember here a[i] and p[i] access same values of array.

---

21) Complete the code of "add()" function, which has to return addition of 5 values in the array.

```
void main()
{   int a[5]={10,20,30,40,50};
    k=add( a );     //  k=add( &a[0] );
    printf("addition of  5  values is  %d", k);
}

int  add( int p[ ]  or  int *p )
{   -----
    -----
}
```

op: addition of 5 values is 150

---

22) Complete the code of "show ()" function, which has to display array of 'N' values in reverse order.

```
void main()
{   int a[20]={10,20,30,40,50,60,70,80 };
    show ( a , 8 );          // here 'a' is array-base address,  and 8 is no.of values in the array
}

void show ( int p[ ] or int *p ,   int  n)
{   -----
    -----
}
```

op: 80 70 60 50 40 30 20 10

---

23) Write a function to search given value in the array is exists or not, let array contained 10 unique values.

```
void main()
{   int  a[10]={11, 34, 88, 49, 15, 66,  45, 23, 32, 17 };
    int key, bool;
    printf( "enter searching value:" );
    scanf( "%d", &key );
    bool=search( &a[0] , 10,  key );          // here 10 is no.of values in the array
    if(bool==1) printf("element exist");
    else printf("element not exist");
}

int search( ----- )          // this function returns bool value
{   ------
    ------
}
```

---

24) Write a function called find (), which takes array base address & number of values in the array and it has to return the **first** +ve and –ve values in the array, if not exist return as 0.

function proto-type is : **void find(int a[] , int n , int *pve , int *nve);**

Later write a main() to check this function with sample values. Complete the following code

```
void main()
{       int  x, y, A[10]={18, 12, 45, -78, 22, 10, 21, 44, 53,10} ;
        find(A,  10, &x, &y);       // x is to hold first +ve,  y is to hold first –ve.
        ----
        ----
}
void find( int A[] , int n , int  *pve , int  *nve )
{       ----
        ----
}
```

---

25) Write a function called findBigSmall(), which takes array base address & number of values in the array and to return the big & small of them. Later write a main() function to check it

```
void main()
{       int   A[10]={18, 12, 45, -78, 22, 10, 21, 44, 53,10} ;
        int  x,y;
        findBigSmall(A,  10, &x, &y);       // x is to hold big ,  y is to hold small.
        printf("big is %d  ,  small is %d ", x, y);
}
void findBigSmall ( int A[] , int n ,  int  *pBig  ,  int  *pSmall )
{           -------
            -------
}
```

Home Work

---

26) Write a function to add 10 grace marks to one subject out of 3 subjects; where such subject marks less than of all other subjects and after adding 10 grace marks, it should not cross 100. (100 is highest marks)

ip: 60 70 90 ip: 70 80 55 ip: 93 97 98
op: 70 70 90 op: 70 80 65 op: 100 97 98

```
void main()
{       int a,b,c;
        scanThree( ----);
        addGraceMarks( ----);
        printf("\n output is: %d  %d  %d", a, b, c);
}
void scanThree( ------ )
{       --------
        --------
}
void addGraceMarks( ------- )
{       --------
        --------            // add marks here
}
```

---

27) Write a function to increment given time by N seconds, where h, m, s and N are argument to function.

```
void main()
{       int  h, m, s, N;
        printf("\n Enter time  :");
        scanf("%d%d%d", &h, &m, &s);
        printf("\n Enter no.of seconds to increment  time  :");
        scanf("%d", &N);
        increment( ------ );
        printf("\n output is %d  %d  %d", h, m, s);

}
void increment( ------- )
{       -------
        --------
}
// sample code for adding 'N' to H,M,S
    N=h*3600 + m*60 + s + N;       // converting total time into seconds and also adding N to it.
    //  distributing  'N'  to  h, m, s format
    h=N/3600;
    m=(N%3600)/60;
    s=(N%3600)%60;
```

---

28) Write a function to increment date by one day, this function takes day, month, and year as arguments and return the incremented date.

```
void main()
{       int d,m,y;
        scanf("%d%d%d", &d, &m, &y);
        increment( ---- );
        printf("\n output: after incrementing the date is %d  %d  %d", d, m, y );
}
void increment( ------- )
{       --------
        --------
}

// sample code for incrementing date for n days( modify this code according to pointer )
        int  arr[13]={0,31,28,31,30,31, …};
        a[2]=28+(y%4==0);
        d++;                    // incrementing day by 1-day
        if( d>arr[m] )          //  after incrementing day, if it crosses month ending point, then shift to next month
        {       d=1;  m++;
                if(m==13)
                {       m=1; y++;
                }
        }
```

---

29) Write a function to copy one array of integer values to another array, this function takes source & destination array base address and number of elements to copy; later write a main() function to check it.

```
void main()
{   int a[20]={12, 45, 78, 22, 10, 21, 44, 53}, n=8;
    int b[20];
    copyArray( b, a, n );   // it is like b=a,  copy all a[] to b[] for 'n' elements
    printf("\n after copying elements, the array b[] is :");
    for(i=0; i<n; i++)
        printf( "%d  ", b[i] );
}
void copyArray( int  b[], int a[], int n)
{   --------
}
```

---

30) Let array A[] contained some values where some values are odds and some are evens.

Now our job is to copy all odds to array B[] and all evens to array C[]. Finally print both arrays.

use A[0] to store count of values in the A[ ] . here actual input values are stored from A[1]

use B[0] to store count of odds in the B[ ]

use C[0] to store count of evens in the C[ ]

```
void main()
{   int A[20]={8, 12, 45, 78, 22, 10, 21, 44, 53} ;   // here first value '8' represents count of array values.
    int B[20], C[20];
    copyOddsAndEvens( A, B, C );
    printf("\n all odds  are :");
    showAll(B);
    printf("\n all evens  are :");
    showAll(C);
}
```

```
void copyOddsAndEvens( …… )
{       ……
}
void showAll( int t[]  )
{   int i;
    for(i=1; i<=t[0];  i++)
        printf("%d   ",  t[i] );
}
```

---

31) Write functions called **read5()** & **write5()**, the function **read5()** scans 5 values from keyboard and stores into array and the function **write5()** prints such 5 values on the screen.

```
 ip: 12  32  43  57  11
op: 12  32  43  57  11
void main()
{   int  a[5];
    read5( --- );
    write5( --- );
}
void read5( --- )
{   ------
    ------
}
void write5( --- )
{   ------
    ------
}
```

---

32) Write functions called **readN()** & **writeN()**, the function **readN()** scans N values from keyboard and assigns into array and the function **writeN()** prints such N values on the screen.

```
void main()
{   int  a[20], N;              // Let us say N<20
    readN(&a[0] , &N );
    writeN(&a[0] , N );
}
void readN( int p[ ] , int *x )
{   printf("enter how many input values::");
    scanf("%d", x);      // x contained &n, so it would be: scanf("%d", &n);
    printf("enter %d values ::", *x);
    for( i=0;  i<*x; i++)
        ----------
}
void writeN( int p[ ] , int N )
{   ------
    ------
}
```

---

33) Let there are two arrays, the A[] contained N1 elements and B[] contained N2 elements.
Now Find B[] is a subset of A[] or not?. If all elements of B[] is exist in A[] then B is said to be subset of A.
Let us assume each independent array contained unique elements without duplicate in it.

*Write a function for scanning input values to array.
*Write a function to search a value in the array
*using these two functions, simplify and write the main program.

```
void main()
{       int a[100], b[100], n1, n2;
        scanArray( a , &n1 );
        scanArray( b , &n2 );
        bool=isSubset( a, n1, b, n2 );
        if( bool==1 ) printf("yes, it is subset");
        else printf("not subset");
}
int search( int a[], int key) { … }
void scanArray( int a[] , int *pn ) { … }
int  isSubset( int a[], int n1,  int b[], int n2 )
{       for(i=0; i<n2; i++)
                if( search( a , b[i] ) == 0 )        // not found, so not subset
                        return 0;
        return 1;
}
```

---

34) Write a program to accept two array of elements (two sets with unique values) and each contained N1, N2 values respectively, let the array names are A[], B[]. now print

1) print A[] Ω B[] (A intersection B , here print common elements of A , B)
2) print A[] – B[] (print elements which are in A but not in B)
3) print A[] U B[] (A union B) → A+B-A // print

note: Write a function for scanning input values to array.
Write a function to search a value in the array (like above program)
using these two functions, simplify and write the program.
Solve using single program or write separate programs for each output.

---

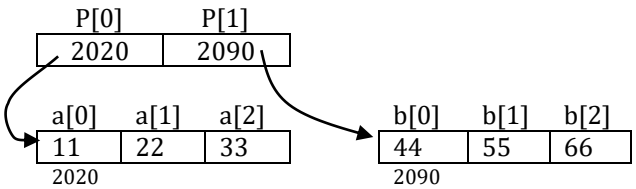
35) In the beginning examples, the swap() fn swaps only given variable values of a specific type, now extend above swap() fn to work for any data type, here swaps byte by byte using **char*** pointer.
This function takes three arguments: address of variable1, variable2 and sizeof(data);


```
// this swap() fn, swaps any type of data (it can be int/float/char/long-int)
   void swap(void *p, void *q, int size);
   {      char *x, *y;
          x=(char*)p;              // type casting, converting  void*  to  char*
          y=(char*)q;
          for(i=0; i<size; i++)
          {      --------
                 --------          // here swap byte by byte using *(x+i) and *(y+i)   or  x[i] or y[i]
                 --------
          }
   }
   void main()
   {      int a=12, b=13;
          float c=12.45, d=45.59;
          swap( &a, &b, sizeof(int) );
          printf("\n after swapping  a, b values are %d %d", a, b );
          swap( &c, &d, sizeof(float) );
          printf("\n after swapping c, d values are %d  %d" , c, d );
   }
```

---

36) Complete the following code to print 2 array values.

```
   void main()
   {  int a[3]={11,22,33},  b[3]={44,55,66};
      int *p[2];
          p[0]=&a[0];
          p[1]=&b[0];
          for(i=0; i<2; i++)
          {     for(j=0; j<3; j++)
                      printf( -----);        // fill yourself to print array values.
                printf("\n");
           }
   }
```



---

37) Write a program to accept a valid date from keyboard and increment it by N days.

ip: (d,m,y) → 1 1 2009

N → 366 (days to increment)

op: 2 1 2010

logic: step1: take loop to repeat N times

step2: increment date by 1-day in every cycle of loop.

step3: after looping, print date.

```
void main()
{       int d,m,y, N;
        here scan  d,m,y and N
        incrementDateByNdays( &d, &m, &y, N );
        printf("date after incrementing %d is %d-%d-%d", N, d, m, y);
}
void  incrementDateByNdays ( int*, int*, int*, int );   // here take loop and call below fn for N times
void  incrementDateBy1day( int*, int*, int* );
```

---

38) Write a program to accept a valid date from keyboard and decrement it by N days.

ip: 1-3-2010 and 366

op: 28-2-2009

---

39) Accept two-dates from keyboard and print their difference in days, let the two dates are valid.

step1: let two dates are date1, date2 [take date1 as → d1, m1, y1 ; take date2 as → d2, m2, y2]

step2: Let date1 < date2, if not then swap them

step2: take loop and repeatedly increment date1 by 1 day until it reached to date2, the logic as

```
while(d1<d2 ||  m1<m2 || y1<y2)
{   count++;         //  to count diff in days
    incrementDateBy1(&d1, &m1, &y1 );
}
printf("difference count is = %d", count);
```

ip: date1 = 1-1-2009

date2 = 2-1-2010

op: difference count is = 366

---

40) Write a program to accept a date from keyboard and find day of the week.

simple logic: Take one fixed date like your birth day; find diff between your birth date and given input date, after finding difference in days, divide it with 7, if remainder is zero then that day is exact day of your birth day, if remainder is 1, that day is next day to your birth day, in this way you can find day of the week. Ensure that your input-date should be greater than birth-date.

ip: 10-5-1973

op: "Thursday"

step1: take fixed like: d1=10,m1=5,y1=1973; // this is my birth day and it is Thursday

step2: take d2,m2,y2 as input date. This must be greater than above (next date)

step3: find difference of two dates. (Let it is count)

step4: count=count%7;

step5: if(count==0) printf("Thursday")

else if(count==1) printf("Friday");

logic: there is a scientific formula to solve this problem in simple way, google it.

---

41) Write a program to accept month and year from keyboard and print calendar of that month.

logic: Using previous program we can solve easily.

Characters & Strings

01) Code to accept a character and find whether it is lower/upper case alphabet or digit or any other special character.

ip: A ip: $
op: upper case alphabet op: special symbol

---

02) Code to accept an alphabet from keyboard and convert to opposite case; if lower case alphabets then convert it into upper-case alphabet and vice-versa.

ip: A ip: a
op: a op: A

---

03) Code to print given character ASCII value on the screen.

ip: A ip: a
op: 65 op: 97

example: char ch='A';
printf("the ASCII value of %c is %d", ch , ch); → the ASCII value of A is 65
%c → prints ASCII symbol, whereas %d → prints ASCII code

---

04) Write a function called **getUpper(),** it returns given alphabet to upper case, later write main() function to test it. For example,

```
void main()
{   char ch='a';
    ch=getUpper(ch);
    printf("upper cae is %c", ch);
}
char getUpper(char ch)
{   ------
    ------
}
```

note: do not use any library function like 'toupper()'

---

05) Following program accepts two numbers from keyboard but each of this number contain only single digit as input and these are in character format, scanned by getchar() or scanf() function, it prints addition of these two numbers, but shows wrong output, fix it.

```
void main()
{   char v1 , v2 ;
    int v3;
    v1=getchar()  or  scanf("%c", &v1);
    v2=getchar()  or  scanf("%c", &v2);
    v3=v1+v2;
    printf("%d ", v3);
}
```

ip: 3 4 // the ASCII values of '3' & '4' characters are 51,52. These values are stored in v1, v2. So output is 103 (wrong)
op: 103

Note: ASCII value of '0' to '9' characters is 48 to 57, to get proper output, subtract ASCII value of '0' from v1&v2 before adding two values. This is as given below

```
v1=v1-'0';      // v1=v1-48
v2=v2-'0';      // v2=v2-48;
v3=v1+v2;
```

---

06) following code shows ASCII symbols of A-Z and a-z

```
for( i=0; i<26; i++)
    printf("%c", 65+i);    or   printf("%c", 'A'+i );      // output is:  ABCDEF ….Z

for( i=0; i<26; i++)
    printf("%c", 97+i);    or   printf("%c", 'a'+i );      // output is:  abcdef …z
```

using above code samples, print following output as shown below (print 6 rows only)

```
ABCDEF
ABCDE
ABCD
ABC
AB
A
```

---

07) Code to accept name of a person and print ASCII value of every character

```
ip: Sri hari
op: S=83, r=144, i=105, space=32, H=72, a=97, r=144, i=105
```

---

08) Code to count number of vowels in a given string

```
ip: all fruits are apples
op: vowel count=7
```

---

09) Code to accept a string from keyboard and count upper and lower case alphabets separately

```
ip:  C-Family
op: Upper case is 2
      Lower case is 5
```

---

10) Code to accept a string from keyboard and convert upper-case alphabets to lower-case and vice-versa

```
ip: C-Family
op: c-fAMILY
```

---

11) Code to accept a string and print in reverse form. (print from last character to first character)

```
ip: srihari
op: irahirs
```

---

12) Code to accept a string and print following way

```
ip:  computer
op1:  computer                output2:  computer
      omputer                           compute
      mputer                            comput
      puter                             compu
      uter                              comp
      ter                               com
      er                                co
      r                                 c
```

---

13) Code to accept a string and print its length

ip: hello

op: 5

method1: try without using function (write total code in main() fn)

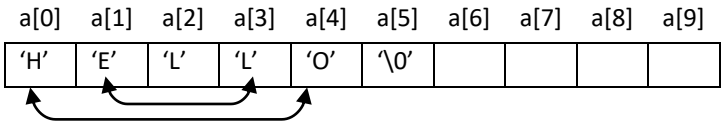
method2: try by writing user-function, the function take string base address as argument and returns length, later check with main() fn, this is as given below

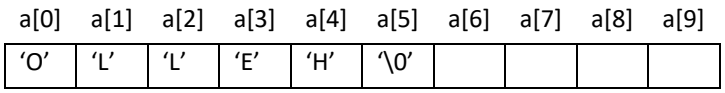
```
void main()
{   char a[20]="computer";
    int k;
    k=myStrlen( &a[0] );
    printf("\n length of computer is: %d", k );      // length of computer is: 8
    k=myStrlen("hello");
    printf("\n length of hello is: %d", k);          // length of hello is: 5
}
int myStrlen( char *p / char p[] )
{   -----
    -----
}
```

---

14) Code to reverse a given input string, later print on the screen

To reverse the string, swap a[0] by a[n-1], a[1] by a[n-2], a[2] by a[n-3],…etc, here 'n' is length of string





Logic: try with/without using function, the code using function is

```
void main()
{   char a[20]="HELLO";
    myStrrev(a);  or myStrrev( &a[0] );
    printf("output is %s", a);  → OLLEH
}
void myStrrev( char *p / char p[] )
{   -------
    -------
}
```

---

15) Write a function to check given string is Palindrome or not? This function returns Boolean value.
(If string and its reverse are equal then it is called Palindrome)

ip: MADAM
op: "yes palindrome"

```
void main()
{   char a[20]="MADAM";
    int bool;
    bool=isPalindrome( a );
    if(bool==1)  printf("yes, palindrome");
    else   printf("no, it is not palindrome");
}
int isPalindrome( char *p )
{   -------
    -------
}
```

---

16) Write a function called myStrlwr(), to convert all upper case alphabets to lower case.

```
void main()
{   char a[20]="ABCdeF";
    myStrlwr(a);
    printf("output is %s", a);  → abcdef
}
void myStrlwr( char *p )
{   -------
    -------
}
```

---

17) Let X , Y are two strings, scan and print whether X>Y or X<Y or X==Y

Note: Strings are compared based on alphabetical order but not on length.

input1: Hello	Input1: Hello	input1: Hell
input2: Hello	input2: Hell	input2: Hello
Output: X==Y	Output: X>Y	Output: X<Y

Logic: Let us try with functions **myStrcmp(),** the function returns as:
case1: if two strings are equal then return(0)
case2: if two strings are not equal then returns ASCII difference of first un-matched characters.
the difference is +ve/-ve. if X>Y then returns +VE, if X<Y then returns -VE, if X==Y then returns 0;

```
void main()
{   char X[20] , Y[20];    int k;
    printf("enter string1:");
    gets(X);
    printf("enter string2:");
    fflush(stdin);
    gets(Y);
    k=myStrcmp( X , Y );
    if(k==0) printf("X==Y");
    else if(k<0) printf("X<Y");
    else printf("X>Y");
}

int myStrcmp( char *p, char *q )
{   --------
    compare char by char, if not equal at any point then stop the loop and return p[i]-q[i]
}
```

---

18) The above program treats, lower case alphabets different from upper case, here "ABCD" is different from "abcd". Extend the above program by ignoring case

Input1: ABCD
Input2: abcd
output: both are equal

hint: before comparing , convert two chars into same upper case for example

```
ch1=toupper( a[i] );
ch2=toupper( b[i] );
if( ch1!=ch2) …
```

---

19) Program to copy a string from one location to another location (one array to another array)

In programming, it is often need to copy a string from one to another location, for this, if one tries to copy a string using assignment operator then compiler shows an error.

For example:

```
char a[20]="hello World";
char b[20];
b=a;          // b=&a[0];
```

Here we are trying to assign &a[0] to 'b', here 'b' is an array not a pointer, therefore it shows an error.

The solution is, copy char-by-char from source to destination array, for example,

```
b[0]=a[0]
b[1]=a[1]
b[2]=a[2] ….Using loop
```

```
void main()
{   char a[20]="hello" ;
    char b[20];
    -----          // here copy char-by-char using loop. For example, b[0]=a[0], b[1]=a[1], b[2]=a[2], …etc
    printf("after copy, the string is %s", b);
}
```

---

try using functions, the code is as given below

```
void main()
{   char a[20]="hello" ;
    char b[20];
    myStrcpy( b , a );      // this is like  b=a
    printf("after copy, the string is %s", b);
}

void  myStrcpy( char  *b , char  *a )
{   -------
    -------
}
```

---

20) Program to accept two strings and append second string at the end of first string

input1: Hello
Input2: World
Output: HelloWorld

logic: try with/without using functions. Using functions, the code is as given below

```
void main()
{   char a[20]="Hello", b[20]="World";
    myStrcat( a , b );
    printf("output is %s", a );   //HelloWorld
}
void myStrcat( char *p, char *q )
{   -------
    -------
}
```

---

21) Code to accept a string, the string may have digits, now print sum of all digits, for example

ip: ABC**9**DEF**8**GH**35**XYZ**6**
op: 9+8+3+5+6 → 31

```
if ( isdigit(a[i])) sum=sum+(a[i]-48);   // 48 is zero ascii code
```

---

22) Code to accept arithmetic expression string like given below, and print sum of all numbers

ip: 123+456+100 // Let the input expression contains only '+' operator with values.
op: total is : 679

```
for( i=v=sum=0;  a[i]!='\0';  i++)
{   if( isdigit(a[i]) )
        v=v*10+(a[i]-48);      //  generating number from ascii codes
    else                       // '+' operator found
    {   sum=sum+v;
        v=0;          // as one number is over
    }
}
sum=sum+v;       // this is for last numeric value.
printf(" sum is %d", sum);
```

---

23) Code to accept a string, the string may have numbers, now print sum of all numbers, for example

ip: ABC**29**DEF**38**GH**135**XYZ**16**
op: 29+38+135+16

---

24) Write a program to accept an arithmetic simple expression from keyboard and print its sum.

The operators can be any + - * / % and the values are integers

ip: 10+20 ip: 20*30 ip: 20/10
op: 30 op: 600 op: 2

---

25) Code to accept a multi word string and print no.of words in the string

(Let the words are separated by single space, also try when more spaces exist)

ip: all apples are fruits but all fruits are not apples
op: 10 // logic is: if(a[i] == ' ') count++;

---

26) Code to accept multiword string and print each word length (Let words are separated by single space)

ip: all apples are fruits

op: 3 6 3 6

hint: if (a[i]==space)

{ then print 'count'

reset 'count' to zero for next word count

}

else count++

---

27) Code to accept a multi word string and convert all first characters of each word to upper-case and remaining to lower case (assume single space between words)

ip: all apples are fruits and all fruits are not apples

op: All Apples Are Fruits And All Fruits Are Not Apples

step1: change first word's first character to upper case like a[0]=toupper(a[0]);

step2: now start loop from 2nd character and check for spaces and convert as

```
for(i=1; a[i]!='\0'; i++)
{   if( a[i-1]==' ')      // if previous character is space, then a[i] is the first character of word
        a[i]=toupper( a[i] );
    else  a[i]=tolower( a[i] );
}
```

step3: print string

---

28) Code to accept a string and print each alphabet frequency

(Let us assume string contains lower case alphabets, remember total alphabets are 26)

ip: all are good programmers

op: a-3 times repeated

d-1 time repeated

e-2 times repeated ….

1. First take array count[] with size 26 and initialize all cells with zero. This is like: **int count[26]={0};**
2. count[0] for 'a', count[1] for 'b', count[2] for 'c', etc. // count[] array stores count of all alphabets
3. Now check each alphabet in the string, if alphabet is 'a' then increment count[0], if 'b' then increment count[1], etc. for this, use following logic

```
int count[26]={0};
for( i=0; a[i]!='\0'; i++)
{   k=a[i]-'a';
    count[k]++;
}
```

4. Finally, print all nonzero cells of count[], how many times each alphabet is repeated

ip: abc aaa bag cad caf

op:

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]	…			
7	2	3	a	0	1	1	0	0	0				
'a'	'b'	'c'	'd'	'e'	'f'	'g'	'h'	'l'	'j'	…			

(if you not followed this logic, refer arrays chapter problems)

---

29) Check if two Strings are anagrams of each other or not?
Two strings are anagrams if they are written using the same exact letters, ignoring space, punctuation, capitalization and order. Each letter should have the same count in both strings. For example, the Army and Mary are an anagram of each other.

ip1: Army ip1: area
ip2: Mary ip2: are
op: Yes, anagrams op: No, not anagrams

hint: use above program logic

---

30) One of the most common string interview questions: Find the first non-repeated (unique) character in a given string, for Example, if given String is "Morning" then it should print 'M'.

ip: Hello hey ip: Madam
op: o op: d

---

31) Code to accept a multi word string and print each word in reverse(assume single space b/w words)
(use following reverseWord() function to reverse the word, using this, we can solve easily)

ip: all apples are fruits
op: lla selppa era stiurf

```
void main()
{   ------
}
void reverseWord(char *s, int  i , int j )     // i & j  are starting & ending index of word
{   char t;
    while( i < j )
    {   t=a[i];
        a[i]=a[j];
        a[j]=t;
        i++;  j--;
    }
}
```

---

32) Program to accept a multi word string and find whether specific sub-string exists or not?

ip: main-string: all apples are fruits
sub-string: are
op: yes

note: this is very important program.

```
for(i=0; a[i]!=0; i++)
{       for( j=0;  j<strlen(b);  j++ )
        {       if( a[i+j]!=b[j] )
                    break;
        }
        // if above j-loop not stopped in the middle by break then substring found, so print it
        if( j==len)
             here stop the i-loop and print as substring found or else continue the i-loop
}
```

---

33) Extension to above program, write a function called myStrstr(), which takes main-string & sub-string as arguments and find first occurrence of sub-string in the main-string, if found return its address in main-string, if not found return NULL.

```
void main()
{   char a[20]=”abcdefghijklmno”;
    char b[20]=”def”;
    char *p;
    p=myStrstr( a , b );        // this function returns “&d” in a[]
    if(p==NULL)  printf(“substring not found”);
    else  printf(“substring found and remaining part of main-string is %s”, p);
}
char* myStrstr( char *m , char *s )
{   -------
    -------
}
```

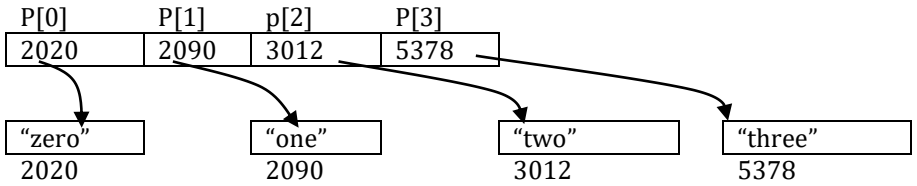
op: substring found and remaining part of main-string is: defghijklmno

---

34) Following example explains how to print given numeric single digit in English words, based on this logic, print for N-digit number.

ip: 3 ip:8
op: three op: eight

```
void main()
{   char *p[] = {"zero", "one","two","three","four","five","six","seven","eight","nine"};
    int n;
    printf("enter any single digit :");
    scanf("%d",&n);
    printf(“%s “,  p[n] );
}
```



Extend above program to print given number in English words as

ip: 3456
op: three four five six.

Also extend above program to print given number in English words as

ip: 3456
op: three thousand four hundred five six.

Home Work

35) Write a program to accept a multi word string and find how many times the given sub-string is repeated
ip: enter main-string: **all apples are fruits but all fruits are not apples**
enter sub-string: **are**
op: 2 times

---

36) Write a program to accept a multi word string and replace a given sub string with new sub string. Consider replacing string & new string lengths are equal
ip: enter main-string: all apples are fruits but all fruits are not apples
enter removing-string: are
enter replacing-string: ###
op: all apples ### fruits but all fruits ### not apples

---

37) Write a program to accept a multi word string and replace a given sub string with new sub string. Consider replacing string & new string lengths may or may not be equal
ip: enter main-string: all fruits are good but some are sour
enter sub-string: are
enter replace-string: were
op: all fruits were good but some were sour

---

38) Write a program to accept a multiword string, and find biggest word length and print it.
ip: all fruits were good but some were sour
op: biggest word=fruits, length=6

---

39) Write a program to accept a text, char by char until user entered '^' as end of input, store all characters into array, later count no.of words, lines, digits …etc;
ip: "this program is on strings↲
and getting command over↲
logic and string manipulations↲
Today date is 25-2-2020" ^
op: words count = 17
Lines count = 3
Digits count=7

---

40) Write a program to accept long integer value from keyboard and print after adding coma symbol in appropriate position. (Hint: convert integer value to string and then add coma symbols)
ip: 2983453
op: 29,83,453

---

41) Write a program to accept a multi word string and words may separate by more than one space, remove all unnecessary spaces between words(more than one space) [this is known as trimming a string]

A	l	l						c	H	A	i	r	s					a	r	e			B	l	u	E	\0

A	l	l		c	h	A	i	r	S		a	r	e		b	l	u	e	\0								

---

42) Write a program to accept a multi word string and words may separate by more than one space, remove all extra spaces between words, extra spaces may happened before & after coma(,) or full stop(.)

T	W	o					A	r	e		r	E	d		,			t	W	o		a	r	E			b	l	u	e	\0

T	W	o		a	R	e		r	e	d	,	T	w	o		a	r	e		b	l	U	e	\0							

---

43) Write a program to sort N string into ascending order; the strings are stored in 2D array as

String1	'O'	'R'	'A'	'N'	'G'	'E'	'\0'				
String2	'B'	'A'	'N'	'A'	'N'	'A'	'\0'				
String3	'G'	'U'	'A'	'V'	'A'	'\0'					
String4	'A'	'P'	'L'	'L'	'E'	'\0'					

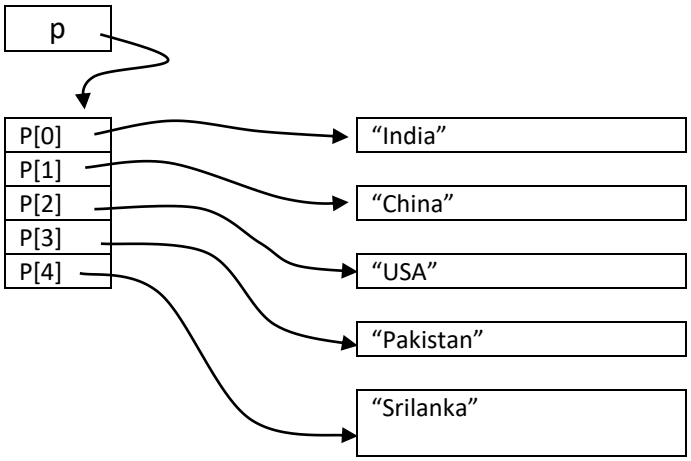
After sorting, the strings are

String1	**'A'**	**'P'**	**'L'**	**'L'**	**'E'**	**'\0'**					
String2	**'B'**	**'A'**	**'N'**	**'A'**	**'N'**	**'A'**	**'\0'**				
String3	**'G'**	**'U'**	**'A'**	**'V'**	**'A'**	**'\0'**					
String4	**'O'**	**'R'**	**'A'**	**'N'**	**'G'**	**'E'**	**'\0'**				

---

44) fill the code to sort N elements in the following program, this program accepts N strings from keyboard, later sorts N strings into ascending order….

```
void main()
{   char  **p=NULL, a[50];
    int N, i;
    printf("enter how many strings :");
    scanf("%d", &N);
    p=(char**) malloc(sizeof(char**)*N );
    for(i=0; i<N; i++) {
        printf("enter string:");
        fflush(stdin);
        gets(a);
        p[i]=(char*)malloc(strlen(a)+1);
        strcpy(p[i], a);      }
    // write code for sorting  (while sorting, if swap is needed then swap addresses not strings)
    --------
    --------
    for(i=0; i<N; i++)
        puts(p[i]);
}
```

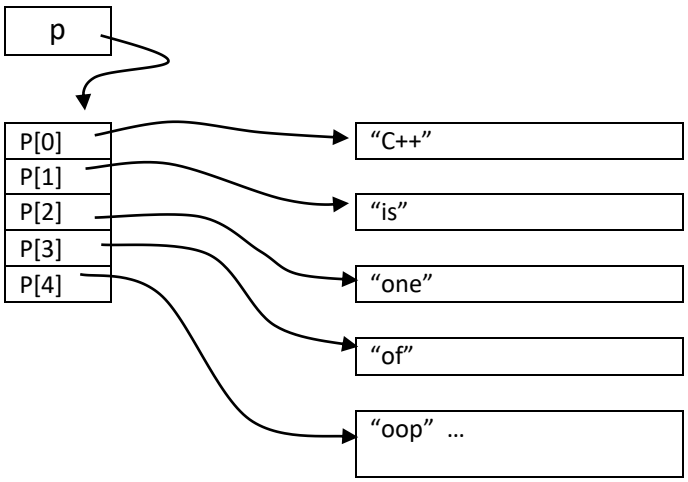


---

45) Write a program to accept a multiword string and break down each word into a separate string and store into separate array of strings as given below

note1: let the words are separated by single space

ip: C++ is one of oop language



Note2: Also try when words are separated by more spaces, coma with/without spaces, full stop, etc (here trim the string before splitting it) , for example, the input string is,

ip: all white are good, all red are damaged ,but all are fruits. Yesterday brought from super market.

Structures

1) write a program to accept student marks like idno, name, marks1, marks2, later find total, average of marks and print them.

Note: try with & without using functions

ip: 101 Srihari 60 70 (idno, name, marks1, marks2)

op: idno=101 , name="Srihari" , marks1=60 , marks2=70 , total=130 , average=65.00

Using functions, the code is as follows

```
struct Student
{   int idno;
    char  name[30];
    int m1, m2, total, avg;
};
void main()
{   struct Student  s;
    scan( &s );                  // scan input for idno, name , marks1, mark2 in this function
    find( &s );                  // find total and average of marks in this function
    show(s);  or show(&s)        // show output in this function ( use either call-by-value   or  call-by-reference )
}
void scan( ---- )
{   ---
    ---
}
void find ( ----)
{   ----
    ----
}
void show ( ----)
{   ----
    ----
}
```

---

2) Write a program to accept radius of circle and find area and perimeter

Note: try with & without using functions

```
struct Circle
{   int radius;
    float area, perimeter;
};
void main()
{       struct Circle  a;
        scan( &a );          // call by reference
        find( &a );          // call by reference
        show( a );   or  show(&a)       // use either call-by-value or call-by-reference
}
void scan( ---- ) { ---  }
void find ( ----) {  ---  }
void show ( ----) { --- }
```

---

3) Write a program to accept two dates and find whether they are equal or not?

Note: try with & without using functions

ip: 12 9 2010 ip: 12 9 2010
12 9 2010 13 9 2011
op: equal op: not equal

Using functions, the code is as follows

```
struct Date
{   int d, m, y ;
};
void main()
{   struct Date a , b;
    int k;
    readDate( &a );              // scan date1 (day, month, year)
    readDate( &b );              // scan date2 (day, month, year)
    k=compareDates( a , b );     // compares two dates and returns 1/0
    if( k==1) printf("equal");
    else printf("not equal");
}
void readDate( ---- )
{   ---
}
int compareDates( ----)
{   ---
}
```

---

4) Write a program to accept date and find whether it is valid or not?

ip: 12 9 2010 ip: 32 9 2010
op: valid op: invalid

Using functions, the code is as follows

```
struct Date
{   int d, m, y ;
};
void main()
{   struct Date  a;
    int bool;
    readDate ( …. );
    bool= isValidDate( …. );
    if( bool==1) printf("valid date");
    else printf("invalid date");
}
void  readDate(…..)
 {   ----------
     ----------
 }
int  isValidDate( .....)
{   ----------
    ----------
}
```

---

5) Write a program to accept two times, let the two times are employee working time in two shifts, later add & print total time worked in two shifts.

ip: 12 50 58 (shift-1 worked duration : Hours=12, Min=50, Seconds=58)
2 53 55 (shift-2 worked duration : H=2, M=53, S=55)
op: 15hr 44min 53sec (total duration worked in two shifts)

```
struct Time
{   int h, m, s ;
};
void main()
{   struct Time a,b,c;
    read( &a );          // scan time1 (hours, minutes, seconds) in this function.
    read( &b )           // scan time2 (hours, minutes, seconds) in this function.
    c=add( a , b );      //  add two times like c=a+b
    write( c );          // print output time on the screen
}
void read( ---- )
{   ---
    ---
}
struct Time add(---  ,  ---)
{   ----
    ----
}
void write( ---- )
{    ----
    ----
}
```

---

6) Code to accept a sample time and increment it by N seconds, here N is also an input.

ip: 12 59 58 (sample time taken from keyboard: Hours=12, Min=59, Seconds=58)
124 (sample no.of seconds to increment: N=124)
op: 13hr 2min 2sec (time after incrementing N seconds)

Note: solve using functions

```
struct Time
{   int h, m, s ;
};
void main()
{   struct Time  t;
    read( &t );                  // scan time1 (hours, minutes, seconds) at this function.
    increment( &t , 3600 );      //  increment  't' by 3600 seconds
    write(t);                    // print time on the screen
}
void read( ---- )
{   ---
}
void write(---)
{   ----
}
void increment(---, ---)
{   ----
}
```

---

7) Write a program to accept date (assume input date is valid) and increment it by N days

ip: 12 9 2010 ip: 12 9 2010
365 (increment days) 10 (increment days)
op: 12 9 2011 op: 22 9 2010

Using functions, the code is as follows

```
struct Date
{   int d, m, y ;
};

void main()
{   struct Date a;
    int n=365;                  // here 'n' is no.of days to increment
    readDate( &a );
    incrementDate( &a , n );
    printDate(a);               // printing date after incrementing it
}
void  readDate(.....)  { ..... }
void  printDate(....)  { .... }
void  incrementDate( .....)
{   // sample code for incrementing date for n days (modify this code with respective to structures )
    int  arr[13]={0,31,28,31,30,31, …};
    a[2]=28+(y%4==0);
    for(i=0;  i<n; i++)
    {   d++;
        if( d>arr[m] )
        {   d=1;  m++;
            if(m==13)
            {   m=1; y++;
                a[2]=28+(y%4==0);
            }
        }
    }
}
```

Nested Structure Example

8) Write a program to accept 2 employee details like idno, name, date of joining, later print both employees joined on same date or not? (Here nested structures used)

```
struct Date
{   int d , m , y;
};
strcut Employee
{   int idno;
    char name[30];
    struct Date jd;
};
```

→

```
Struct Employee
{   int   idno
    char  name[30];
    struct Date
    {   int d, m , y;
    } jd ;
};
```

```
void main()
{       struct Employee e1 , e2 ;
        int k;
        read( &e1 );
        read( &e2 );
        k=compare(e1 , e2);
        if(k==0) printf("equal");
        else printf("not equal");
}
void read(---)
{       --------
}
int compare(--- , ---)
{
        --------
}
```

Array of structures

9) Write a program to accept 3 student details such as idno, marks1, marks2 from keyboard and print total & average of marks. Do with and without functions.

```
ip:  101   70   70
     102   80   90
     101   75   75

op: idno    m1     m2     total     avg
    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     101     70     70     140      70
     102     80     90     170      85
     103     75     75     150      75
```

```
struct Student
{       int idno, m1, m2, total;
};
void main()
{       struct Student s[3];       // for 3 students
        -------
}
```

the array of structures and its sample data as shown below

S[0]				S[1]				S[2]			
101	70	70	140	102	80	90	170	103	75	75	150
s[0].Idno	s[0].m1	s[0].m2	s[0].total	s[1].Idno	s[1].m1	s[1].m2	s[1].total	s[2].Idno	s[2].m1	s[2].m2	s[2].total

```
-----
-----
-----
```

Home Work

---

10) This program about two players of the game, where add score, print score, etc.
Complete the code of following functions.

```
struct Game
{       char  *name1,  *name1;    // to hold  players name
        int score1,  score2;      //  to hold two players score
};
void accpetNames()
{       struct Game  g;
        initializeGame(&g  , “Ram”,  “Ravi”);  //  set name of players and also make zero to score1,score2
        addScore(&g, “Ram”, 3);          // 3 is points to Ram
        addScore(&g, “Ram”, 5);
        addScore(&g, “Ravi”, 2);
        showScore( g );                  // display score of two players along with names
}
void intializeGame(struct Game *p, char *x, char *y)
{     p->name1=x;
      p->name1=y;
     p->score1=p->score2=0;
}
void addScore( … ) { … }
void showScore( … ) { … }
```

---

11) Write a program to accept 2 matrices data of size r x c (Let r<10, c<10) and print addition of them.

```
struct Matrix
{   int a[10][10];
    int r , c ;
};
void main()
{   struct Matrix  a , b , c ;
    read( &a );
    read( &b );
    k=add( &c , a , b );
    if(k==0) printf(“Error, two matrices have different sizes” );
    else write( c );
}
void read( ---- )
 {     ---
 }
int add( ---- )
{     ---
}
void write( ---- )
{     ---
}
```

---

12) Write a program to handle list of values like insert, delete, search, display, etc

```
struct List
{   int  *a;                    // pointer to dynamic array to hold values.
    int listSize , count;       // listSize is size of list(size of array),  count is no.of elements in the array
};
typedef  struct List*   List;

void main()
{       List   p;
        p=createList(10);       // 10 is size of list(array)
        insert( p, 12);
        insert( p, 18);
        insert( p, 24);
        insert( p, 35);
        show(p);                //  12, 18, 24, 35
        delete(p,24);
        show(p);        // 12, 18, 35
        if( search( p , 18) == 0 )  printf("element not found");
        else   printf("element found ");
    }

List createList(int size)
{       List p;
        p=(List)malloc(sizeof(*p));
        p->count=0;         // initially no element in Q
        p->listSize=size;
        p->a=(int*)malloc(sizeof(sizeof(int)*size);
        return p;
}

void insert( List p, int value)
{       if( p->count==p->listSize)
        {       printf("error, list is full"); return; }
        p->a[p->count]=value;
        p->count++;     // because one element added to list, so increase count
}

void show( List p) { … }
int  search( List p, int value) { … }
void delete( List p, int value) { … }
```

---

13) Code to accept two complex numbers and print their sum using functions

```
ip: 3   4    ( 3 + 4i )
    7   3    ( 7 + 3i )
op: 10 + 7i

    struct Complex
    {    int real , img;
    };
    -----
    -----
```

14) Let our database contained 10 person's name with phone number, write a program to search phone number using name and vice-verse. Here the database of 10 persons details are initialized in array of structures and assume name & phone are unique (no duplicates)

ip: Ravi ip: 9440 ip: Laxmi ip: Lavanya ip: 9442
op: 9440 op: Ravi op: 7412 op: not exist op: Ramu

```
struct Person
{   char  name[30];
    char  phone[15];
};

void main()
{   char str[30], *x;
     struct Person p[10]={ {"Ravi", "9440"} ,
                           {"Ramu", "9442"} ,
                           {"Lasya",  "8500"} ,
                           {"Laxmi",  "7412"},
                            ...
                         } ;
// non-stop loop to read strings one by one until "end" is pressed, and prints name/phone if found.
      while(1)
      {   printf("enter name or phone:");
          gets( str );
          if (strcmp(str, "end")) break;
          x=find( p , str );
          if(x==NULL)
                printf("not found in the database");
          else
               printf(" %s  is found ", p);
      }
 // this following fn searches for record, if found returns string address or else returns NULL;
      char*  find( ---  , ---)
      {   ------
          ------
      }
```

---

15) Program to add two polynomials using array of structures

$5x^7+14x^5+3x^2+10x^1+18$

$15x^4+10x^3+13x^2+7$

the two expressions are stored in array of structures as given below

p[0]		p[1]		p[2]		p[3]		P[4]		P[5]	
5	7	14	5	3	2	10	1	18	0	0	
coeff	Expo	coeff	expo	coeff	expo	coeff	Expo	coeff	expo	coeff (end)	

p[0]		p[1]		p[2]		p[3]		P[4]	
15	4	10	3	13	2	7	0	0	
coeff	Expo	Coeff	expo	coeff	expo	coeff	expo	coeff	

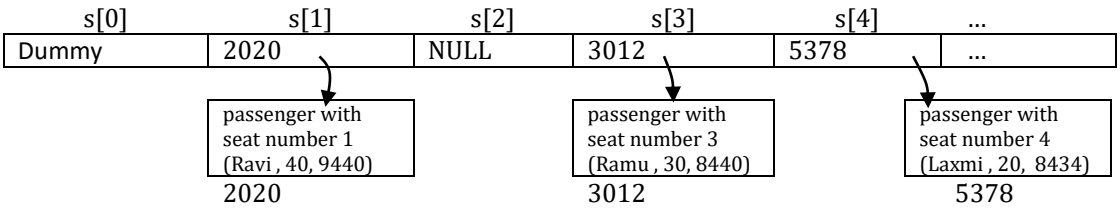
```
struct Poly
{   int  coeff;
    int  exp;
};
```

```
void main()
{       struct Poly p1[10], p2[10], p3[10];
        readPoly(p1);        // accepting first polynomial
        readPoly(p2);        // accepting second polynomial
        addPoly(p3 , p1 , p2);    //  p3=p1+p2
        printf("\n The result of polynomial after addition is\n");
        printPoly(p3);
}
void  readPoly( struct Poly p[ ] )
{   int i=0;
    while(1)
    {   printf("\nEnter coefficient ( 0 to end :");
        scanf( "%d",&p[i].coeff );
        if( p[i].coeff == 0 ) break;                    // stop when coeff is zero
        printf("\nEnter exponent: ");
        scanf("%d",&p[i].exp);
        i++;
    }
}
void addPoly ( struct Poly p3[ ] ,  struct Poly p1[ ] ,  struct Poly p2[ ] )
{       -------
        -------
}
void printPoly ( struct Poly p[ ] )
{       -------
        -------
}
```

---

16) Let us simulate bus reservation problem, here reserve/cancel/print seats according to passenger requirement, the passenger details are: name, age, phone and seat-number is nothing but index of array. the reservation data holds in the array as given below

```
struct Passenger
{   char name[30];
    int age;
    char phone[15];
};
```




```
void main()
{   struct  Passenger  *arr[21]={NULL};      // let total seats are 20 (first is dummy)
    int choice;
    while(1)
    {   printf("\n 1. Reserve ");
        printf("\n 2. Cancel ");
        printf("\n 3. Print ");
        printf("\n 0. Exit ");
        scanf("%d", &choice);                  // stops when seat number is zero.
        if( choice==0) break;
```

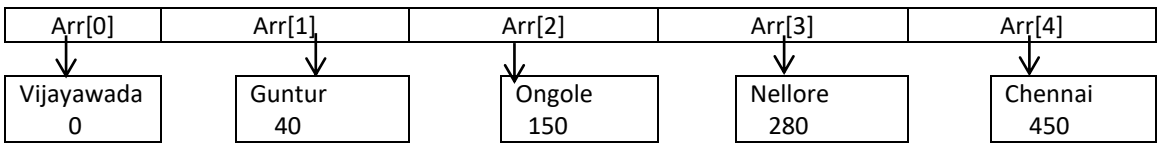
```
        switch( choice )
        {   case 1:   reserve(arr);
                      break;
            case 2:   cancel(arr);
                      break;
            case 3:   print(arr);
                      break;
            case 0:  return;
        }
    }
}

void reserve( struct Passenger *arr[]  )
{   struct Passenger  *p;
    printf("enter seat number(1-20): ");
    scanf("%d", &seatNo);
    if(arr[ seatNo ] != NULL)
    {  printf("sorry seat already reserved");
       return;
    }
    p=malloc(---);          // dynamically creating space for seat
    scanf("%s%d%s",  &p->name, &p->age, &p->phone);
    arr[seatNo]=p;
}

void cancel ( struct Passenger *arr[] )
{       -----
        -----
        // use free() function to delete seat-no
}

void print ( struct Passenger *arr[]  )
{       -----
        -----
}
```

---

17) Write a program to manage Bus Route map like add, delete, search, print, …etc;
for example single route like:: Vijayawada → Guntur → Ongole → Nellore → Chennai

Here we have to maintain details like station-name, distance from origin, distance from previous station the sample structure and main() fn with data as given below

```
struct Point
{   char pointName[30];
    int distanceFromOrigin;
};
```



```
int main()
{   struct Point *arr[10]={NULL};    // let us say maximum points in a route is 10
    addPoint( arr , “null”, “Vijayawada”, 0 );
    addPoint( arr , “Vijayawada, “Guntur”, 40 );
    addPoint( arr , “Guntur”, “Ongole” , 150 );
    addPoint( arr , “Ongole”, “Nellore” , 280 );
    addPoint( arr , “Nellore”, “Chennai” , 450 );
    showRoute( arr );  [ Vij , 0 ]→[ Gun , 40 ]→[ Ong , 150 ]→[ Nell , 280 ]→[ Che , 450 ]
    deletePoint( arr, “Ongole” );
    showRoute( arr );  [Vij,0]→[Gun,40]→[Nell,280]→[Che,450]
    addPoint( arr , "Guntur","Ongole", 110 );
    showRoute( arr );   [Vij,0]→[Gun,40]→[Ong,150]→[Nell,280]→[Che,450]
    searchRoute( arr, “Guntur”, “Chennai”); [Gun,0]→[Ong,110]→[Nell,240]→[Che,410]
}
void addPoint( struct Point *arr[], char *from, char *to, int distance) { ----- }
void deletePoint( struct Point *arr[], char *pointName)  {  -----  }
void searchRoute( struct Point *arr[], char *from, char *to){  ----- }
void showRoute(struct Point *arr[] )   { ----- }
```

Files

01) This demo program explains how to read & write integers to text file

This program writes 5 integers to file using loop, later read back and shows on the screen.

```
#include<stdio.h>
void main()
{       writeNumbersToFile();
        readBackFromFileAndShow();
}
void writeNumbersToFile()
{       int i;
        FILE *fp;
        fp=fopen("sample.txt", "wt");
        for(i=0;i<5; i++)                   // writing to file: 20   21    22   23   24
            fprintf(fp, "%dLJ", 20+i );     //  LJ → Here it is space bar
        fclose(fp);
}
void readBackFromFileAndShow()
{       int k , n;
        FILE *fp;
        fp=fopen("sample.txt", "rt");
        while(1)
        {       k=fscanf(fp, "%d", &n);         // reading integers one by one from file
                if(k<=0)  break;                // fscanf() returns 0 when end of file is reached
                    printf("%dLJ", n );         // printing: 20   21    22   23   24
        }
        fclose(fp);
}
```

---

02) Same as above program, but binary file organization

in computer, either data/program files are stored only in two formats **A)** text-format **B)** binary-format

there is no other format except these two formats, whether file is pdf,mp3,mp4,doc, .exe, .o ….etc.

A) What is text format?

A) Here data is written in string format (ascii code of each character), for example the number 1234 is stored in ascii codes as 49, 50, 51, 52 (Remember, these ascii codes again stored each in binary values)

B) At end of text file, 26 is inserted as end-of-file mark, this is automatically inserted by fclose() fn.

C) the new line character(\n) is stored in two values \r\n (Unix OS format), but while reading back, it read as single character '\n' in DOS/Windows system, but in Unix OS, it read as two values.

D) We don't worry about how to write & read in string format, these things automatically done by library functions such as fprintf(), fscanf(), fputs(), fgets(), fputc(), fgetc() ..etc.

note: this format is old and now a days this is using very rarely in some places like xml, json files systems.

B) What is Binary format?

b1) Here data is written in the form of structures & objects, that is, as it is in program variables.

For example, the N=1234 is stored as it is in 2-byte(int) format of N.

for this, we use fread(), fwrite() functions, no other functions work.

b2) for binary files, computer doesn't insert **end-of-file-mark**, based on file size the end-of-file is identified.

Following example explains, how to work with binary files (above program in binary format)

```
#include<stdio.h>
void main()
{       writeNumbersToFile();
        readBackFromFileAndShow();
}
void writeNumbersToFile()
{       FILE *fp;   int i, n=20;
        fp=fopen("sample.txt", "wb");
        for(i=0;i<5;  i++, n++)
            fwrite( &n, sizeof(int), 1, fp );      // for binary files, use only fwrite(), fread()
        fclose(fp);
}
void readBackFromFileAndShow()
{       int k , n;   FILE *fp;
        fp=fopen("sample.txt", "rb");
        while(1)
        {       k=fread(&n, sizeof(n), 1, fp);
                if(k<=0)  break;                    //  fread() returns 0 when end of file is reached
                    printf("%dLJ", n );             // printing on the screen: 20   21    22   23   24
        }
        fclose(fp);
}
```

fwrite() syntax ➔ fwrite(&varaible, sizeof(variable), no.of variable items, file-pointer)

&variable → this gives address of variable where the data exist
sizeof(variable) → this gives size of data to be written to file.
no.of variable items → in case array of items exist.
file-pointer → refers pointer to file.

---

03) Let our text file **"input.txt"** contains collections of integers in text format, now read numbers one by one from file and count number of **odds** exist in that file. Let the file contained integers as given below.

file name: "input.txt"
12 17 20 13 29 30 32 35 40
43 27 20 21 29 31 39 11 12
10 8 2 1 29

op: odds count=13

hint: A) to read integers from file, use fscanf(fp, "%d", &n);

B) to create "input.txt" file with numbers, first open "notepad APP", where open a new file with name "input.txt" and type some numbers and save it. (Saving folder must be current working folder)

```
#include<stdio.h>
void main()
{       FILE *fp;  int  n,k;
        fp=fopen("input.txt", "rt");
        if( fp==NULL)
        {   printf("file not found");
            return;
        }
        while(1)
        {   k=fscanf(fp, "%d", &n);      //  fscanf()  returns 0  when end-of-file reached
            if(k<=0) break;              //  if k is zero, then end of file is reached, so stop the loop
            -----
            -----
```

---

04) Let our text file **"input.txt"** contained some numbers, now read numbers one by one from file and copy only **odd** numbers into another file called **"output.txt"**.

file name: "input.txt"
12 17 20 13 29 30 32 35 40 43 27 20 21 29 31 39 11 12 10 8 2 3

→

File name:"output.txt"
17 13 29 35 43 27 21 29 31 39 11 3

```
#include<stdio.h>
void main()
{       FILE *f1, *f2;
        int  n;
        f1=fopen("input.txt", "rt");            // opening for reading
        f2=fopen("input.txt", "wt");            // opening for writing
        if( f1==NULL || f2==NULL )
        {   printf("file(s) not opened");
            return;
        }
        here take infinite loop
        here read an integer from  input.txt  using file pointer "f1"
        if  file  pointer 'f1' reached to end-of-file then stop the loop.
        now write integer to output file using pointer file "f2"
        after loop, close all files.
        printf("successfully copied all odd numbers");
}
```

output: after running this program, checkout odds in the file "output.txt" by opening in "notepad APP"

---

05) Write a program to count number of words, lines, digits in a given text file.

Let the file contents as given below

File name: **"sample.txt"**
one two three 123 four five six 456 seven eight nine 789

op: words count = 12
lines count = 3
digits count = 9

step1: read char by char from file like: k=fscanf(fp, "%c", &ch); // here 'ch' is char-variable
step2: if end of file then stop the loop. like: if(k<=0) break;
step3: if(ch==' ') wordsCount++; (let there is only sinle space exist between words)
step4: if(ch=='\n') wordsCount++ and linesCount++;

---

06) this example explains how to read 'text' from keyboard and writing to file, later read back and shows text content on the screen. We know text means, a group of words and sentences, here this program reads text as **char by char** until user pressed at end control+Z for termination, after reading each char, writes to text file called "sample.txt", later prints such file content on the screen.

```
#include<stdio.h>
void main()
{       readFromKeyboardAndWriteToFile();
        readBackFromFileAndShowOnTheScreen();
}
void readFromKeyboardAndWriteToFile()
{       FILE *fp;    char ch;
        fp=fopen("sample.txt", "wt");
        printf("enter text (at end type F6 key) :");      // F6 or ctrl+Z  used for end-of-input
        while(1)
        {       ch=getchar();              // scanning char by char from keyboard
                if(ch==-1) break;          // getchar() returns -1  when user pressed F6  or ctrl+Z
                fprintf(fp, "%c", ch);     // writing to file
        }
        fclose(fp);   // saving on disk
}
void readBackFromFileAndShowOnTheScreen()
{       int k;  char ch;   FILE *fp;
        fp=fopen("sample.txt", "rt");
        while(1)
        {       k=fscanf(fp, "%c", &ch);          // reading char from file
                if(k<=0) break;                   // fscanf() returns -1 when end of file is reached
                printf("%c", ch);                 // display char on the screen
        }
        fclose(fp);
}
```

---

07) Write a program to copy one file content to another file

note: the binary file organization works for both text/binary format files

```
FILE *fs, *ft;
printf("enter source & destination file names :");
gets(fname1);
gets(fname2)
fs=fopen( fname1,"rb");
ft=fopen( fname1,"wb");
while(1)
{   k=fread( &ch, sizeof(ch),1,fs);
    if(k<=0)break;
    fwrite( &ch, sizeof(ch), 1, ft);
}
fclose(fs);
fclose(ft);
```

---

08) Write a program to accept student details and process them. (use binary file system)
Accept student details from keyboard and write to file called "input.dat", later read back, find total, average, result and store in a separate file called "output.dat", finally print "output.dat" on the screen as

idno	name	marks1	marks2	marks3	total	average	result
101	Srihari	55	65	76	196	65	first class
102	Srinu	30	60	60	150	50	failed

If any subject < 35 then take result="failed"
If passed then print result based on average
If average >= 60 then result="first class"
If average >=50 and <60 then result="second class"
If average <50 then result="third class"

Let the "input.dat" & "output.dat" files as given below

File name: "input.dat"
101 Srihari 55 65 76
102 Srinu 30 60 60
103 laxmi 80 80 70

→

File name: "output.dat"
101 Srihari 55 65 76 196 65 "first class"
102 Srinu 30 60 60 150 50 "failed"
103 laxmi 50 45 55 150 50 "second

```
void main()
{  readFromKeyboardAndWriteToFile();
   processResult();      // "input.dat" → "output.dat"
   showOutputFile();
}
void  readFromKeyboardAndWriteToFile()
{   ---
    // here scan student records one by one until 'idno' is zero.
    ---
}
void  processResult()
 {   ---
     ---
}
void  showOutputFile()
{   ---
    ---
}
```

Home Work

09) Write a program to remove the comment lines in "C" program code file. Comment is : /* ------ */

'C' code file with comments		**After removing comments, the file is**
/* written by srihari,9440-030405 */ void main() { in a=10, b=20,c; /* adding two numbers and printing */ c=a+b; printf("output = %d", c); }	→	void main() { in a=10, b=20,c; c=a+b; printf("output = %d", c); }

---

10) Some programmers write code in awkward (not free readable style), now write a program to rearrange the "C" program code file into readable format. for example

'C' code file with awkward style		**After processing readable style, the file is**
void main() { in a,b,c; printf("enter two values"); scanf("%d%d", &a, &b); c=a+b; printf("output = %d", c); }	→	void main() { in a,b,c; printf("enter two values"); scanf("%d%d", &a, &b); c=a+b; printf("output = %d", c); } 1. start every instruction in new line 2. instructions in the block {} must be appeared inside the block with tab space

---

11) Write a program to count number of keywords in a given "C" program file.
For example the keywords are "if", "else" , "while", "for", ….. etc
for example, the input file "big.c" as given below

File name "big.c"	**Input & output**
void main() { int a, b, c; printf("enter three values:"); scanf("%d%d%d", &a, &b, &c); if(a>b && a>c) printf("big=%d", a); else if(b>c) printf("big=%d", b); else printf("big=%d", b); }	ip: file-name "big.c" op: "if" → found 2 times "else" → found 2 times "while" → found 0 times "for" → found 0 times -----

---

A menu driven program to handle employee records

This program manages employee information: it supports adding newly joined employee details, deleting relieving employee record, modifying address or any other information of employee, printing particulars. Employee details are stored in a separate file called "emp.dat".

This menu run program provides the user to select his choice of operation.

```
        Menu Run
    ------------------------------
    1. Adding new employee
    2. Deleting employee record
    3. Modifying existing record
    4.Printing employee details
    0. Exit
        Enter choice [1,2,3,4,0]:
    #include<stdio.h>
    typedef  struct                 // structure of an employee
     {  int  empNo;
        char name[30];
        float salary;
    }EMP;
    void append(); void modify(); void delete(); void print();      // fn proto-types
    void main()
    {   int choice;
        while(1)                    // loop to display menu continuously
        {   printf("\n\n ================================================================");
            printf("\n 1.append \n 2.delete \n 3.modify\n 4 print \n 0.exit");
            printf("\n   Enter choice [1,2,3,4,0]:");
            scanf("%d", &choice);
            switch(choice)
            {   case 1:  append(); break;
                case 2:  delete(); break;
                case 3:  modify(); break;
                case 4:  print(); break;
                case 0:  exit(0);
            }
        }
    }
// ------------------------------------------------------------------------------------------------------------------
//  this append function appends a record in the file
    void append()
    {   FILE *fp;   EMP e;
        fp=fopen("emp.dat", "ab");
        if(fp==NULL)
        {   printf("\nUnable to open emp.dat file");
            return;
        }
        printf("\n enter employee no:");
        scanf("%d",&e.empNo);
        printf("Enter employee name:");
        fflush(stdin);
        gets(e.name);
        printf("enter salary :");
        scanf("%f",&e.salary);
        fwrite(&e,sizeof(e),1,fp);
```

```
        fclose(fp);
        printf("\n successfully record added");
    }

// ------------------------------------------------------------------------------------------------------------------------
// deleting record, generally, it is not possible to delete a record in a file. Alternative is, copying all records
// into another temporary file except deleting record; later temporary file will be renamed with the original
// file name.
    void delete()
    {   FILE *fp,*ft;   EMP  e;
        int eno, found=0, k;
        fp=fopen("emp.dat", "rb");
        ft=fopen("temp.dat", "wb");
        if(fp==NULL || ft==NULL )
        {   printf("\nunable to open file");
            fclose(fp);  fclose(ft);  return;
        }
        printf("\nenter employee number to delete record :");
        scanf("%d",&eno);
        while(1)
        {   k=fread(&e,sizeof(e),1,fp);
            if(k==0)break;
            if(eno==e.empNo)
                found=1;    // record is found
            else
                fwrite(&e,sizeof(e), 1 ,ft);
        }
        if(found==1) printf("\nRecord deleted success fully");
        else  printf("\nRecord Not found");
        fclose(fp);   fclose(ft);
        remove("emp.dat");
        rename("temp.dat","emp.dat");
    }
// ------------------------------------------------------------------------------------------------------------------------
// Modifying data in a record. First, it searches for modifying record in a file, if record is not found then
// displays error message & returns. If found, then old-salary overwrites by new-salary of a record
    void modify()
    {   EMP e;
        int found=0 , eno, k;
        long int pos;
        FILE *fp;
        fp=fopen("emp.dat","rb+");
        if(fp==NULL)
        {   printf("\nfile not found ");
            exit(0);
        }
        printf("\n enter employee number:");
        scanf("%d",&eno);
        while(1)
        {   k=fread(&e,sizeof(e),1,fp);
            if(k==0) break;
            if(eno==e.empNo)
            {   found=1;    // record is found
                break;
```

```
            }
        }
        if(found==0)
        {   printf("\n Record not found");
            return;
        }
        pos=ftell(fp);                  //  ftell()  gives current position of file pointer
        pos=pos-sizeof(e);
        fseek(fp, pos, SEEK_SET);       // move the file pointer one position back
        printf("old salary : %.2f", e.salary);
       printf("enter new salary:");
        scanf("%f", &e.salary);
        fwrite(&e,sizeof(e), 1 ,fp);      // overwriting old record
       fclose(fp);
       printf("\n address suceessfully modified");
   }

//  ------------------------------------------------------------------------------------------------------------------------------------
// print function, prints all records
    void print()
    {   EMP e;  int k, count=0;
        FILE *fp;
        fp=fopen("emp.dat", "rb");
        if(fp==NULL)
        {   printf("\nfile not found ");
            return;
        }
        while(1)
        {   k=fread(&e,sizeof(e), 1,fp);
            if(k==0) break;
         printf("\n employee number: %d",e.empNo);
         printf("\n employee name : %s",e.name);
         printf("\n Salary: %.2f", e.salary);
         count++;
         printf("\n--------------------------------");
      }
      printf("\n(%d) records found", count);
      fclose(fp);
    }
```