Self Practice Exercises

Beginners C-Programs list

This programs list & c-material softcopy is available in pdfhost.io website, to download this file, search through keyword 'cfamily1999' or "cfamily"

Since 1999

C-Family Computer Education

Flyover Pillar No:16, Service Road, Benz Circle
Vijayawada, AP, India, pincode-520010
9440-030405, 8500-117118

.

Basic Programs

1) Demo program addition of two int-values

```
#include<stdio.h>
void main()
{   int  A,B,C;        // here space creates for A,B,C. each takes 2byte
    A=10;              // assigning 10 to A
    B=20;              // assigning 20 to B
    C=A+B;             // adding A,B and assigning result 30 to C
    printf("output is %d ",  C );     // to substitute int-value in output-string use %d
}                                     // in this case '%' does not work like remainder operator
```

2) Demo program addition of two float-values

```
#include<stdio.h>
void main()
{   float  A,B,C;
    A=10.5;
    B=20.5;
    C=A+B;
    printf("output is %f ",  C );     // for float-type values use %f
}
```

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

Expect the output of following program

```
#include<stdio.h>
void main()
{   int  A,B;          // here memory space creates for A and B, each takes 2 bytes. (total 4 bytes)
    A=100;             // the operator(=) puts 100  into  A
    B=A;               // the operator(=) copies A to B, so after copying, B gains same value of A
    printf("A is %d, B is %d" ,  A ,  B);
}
```

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

```
#include<stdio.h>
 void main()
{   int A=10, B=20, C;
    A=A+1;
    printf("A value  is %d",  A );
    A+1;                              // here A+1 is not assigning to any variable, so it ignores
    printf("A value  is %d",  A );
    A=B=C=200;                        // copies 200 into A, B, C
    printf("A,B,C values are  %d  %d %d",  A, B, C );
}
```

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

ip: if x=3

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

```
#include<stdio.h>
void main()
{   int  x , result;
    printf("enter x value :");
    scanf("%d", &x);
    result=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",  result );
}
```

note: x^3 can be written like above said, but for bigger values like x^{10} use pow() function,this is shown below

6) Dem o 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 ($\sqrt{x}$)

```
#include<stdio.h>
#include<math.h>     // when we use pow() or sqrt() function, we have to  include "math.h"
void main()
{       int  A , B;
        A=pow(2,3);
        B=sqrt(16);
        printf(" A=%d  ,  B=%d", A , B);      //  output:  A=8 , B=4
}
```

7) Demo program how to sum up values one by one.

```
void main()
{       int   sum=0;
        sum=sum+3;
        sum=sum+4;
        sum=sum+5;
        printf("sum is %d ", sum);
}
```

8) Finding Fahrenheit from given Celsius. (formula is: F = 9.0/5*C+32)

This program scans Celsius from keyboard and prints Fahrenheit as output.

ip: enter Celsius: 38

op: Fahrenheit is: 100.4

```
#include<stdio.h>
void main()
{       float   C , F ;
        printf("Enter Celsius:");
        scanf("%f", &C);
        F=9.0/5*C+32;
        printf("Fahrenheit is: %f", F);
}
```

9) Finding area and circumference of circle

The following program accepts radius from keyboard and prints area and circumference.

Logic: based on input value **radius**, the output (area and circumference) is calculated;

Process

input: radius(r)

output: area ($\pi\ r^2$)

circumference ($2\ \pi\ r$)

Input & output

ip: Enter radius of a circle: 5

op: Area = 78.57143

Circumference = 31.40

Program

```
#include<stdio.h>
 void main()
{    float radius, area, circum;
     printf("Enter radius of circle :");
     scanf("%f", &radius);
     area=3.14 * radius * radius;
     circum=2 * 3.14 * radius;
     printf("the Area is %f", area);
     printf("\nthe  Circumference is %f", circum);
}
```

10) Demo program how to sum up input values one by one.

```
void main()
{   int N,sum;
    printf(“enter value1:”);
    scanf(“%d”, &N);
    sum=N;
    printf(“enter value2:”);
    scanf(“%d”, &N);
    sum=sum+N;
    printf(“enter value3:”);
    scanf(“%d”, &N);
    sum=sum+N;
    printf(“sum is %d”,  sum );
}
```

input: 10

input: 20

input: 30

11) Demo program how to add digits in a 3-digit number

The operator ‘%’ gives remainder of division. The operator ‘/’ gives quotient of division. For example

234%10 → 4	234/10 → 23
234%100 →34	234/100 → 2
234%1000 →234	234/1000 → 0
27%10 →7	27/10 → 2
7%10 →7	7/10 → 0

```
void main()
{   N=234;                  // 234 is 3-digit number
    sum=N%10;               // sum=4        (234%10 → 4 )
    N=N/10;                 // N=23         ( 234/10 → 23 )
    sum=sum+N%10;           // sum=4+3
    N=N/10;                 // N=2
    sum=sum+N%10;           // sum=4+3+2    ( 2%10 → 2 )
    printf(“sum is %d”, sum);
}
```

12) Demo program result of different operators

```
void main()
{   int A=10, B=20;
    printf("\n result is %d",  A+B);
    printf("\n %d  +  %d  =  %d", A, B, A+B);
    printf("\n %d  -  %d  =  %d", A, B, A-B);
    printf("\n %d  <  %d  =  %d", A, B, A<B);          A<B → 10<20 → true → 1
    printf("\n %d  >  %d  =  %d", A, B, A>B);          A>B → 20>10 → false → 0
    printf("\n %d  ==  %d  =  %d", A, B, A==B);        A==B → 10==20 → false → 0
}
```

output: 10 + 20 = 30
10 - 20 = -10
10 < 20 = 1 …

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

```
void main()
{   int A=9, B=9;
    A++;
    ++B;
    printf("%d  %d ",  A  ,  B  );
    B = ++A * 2;
    printf("%d  %d ",  A  ,  B  );
    B = A++ * 2;
    printf("%d  %d ",  A  ,  B  );
}
```

++A (pre-increment means, do before
B=A*2; all other operators in this expression

B=A*2;
A++; (post-increment means, do after all other)

Syntax of printf() statement

printf("format string", list of values); → here the "Format string" represents, what format we want to display with the list of values. The format strings like: %d , %5d , %05d , %f , %.2f %05.2f, etc.

%5d → here 5 is said to be maximum width of output value which we want to show on the screen.
if width > output-value size then spaces added by the printf() statement.
if width < digits in value then prints normally. For example,

printf("%7d", 123); → BBBB123 // Here B means blank space, added by printf()

printf("%07d", 123); → 0000123 // pads with zeros instead of spaces

printf("%02d", 1234); → 1234 // here width < digits in value, so prints normally

printf("%7.5f", 123.4); → BBBBB23.40000

printf("%.2f", 2678.4256); → 2678.42

printf("A=%d , B=%d", 10,20); → A=10 , B=20

printf("hello is %d, world is %d", 10, 20); → ?

printf("ABC%dCDE", 222); → ?

printf("A * B = %d" , 10); → ?

printf("%d * %d = %d" , 10, 20, 200); → ?

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

16)
```
A=100;
A=A/2;
printf("%d ", A );
A=A/2;
printf("%d ", A );
```

17)
```
float  A;
A=5/2;
printf(" %f ",  A );
A=5.0/2;
printf(" %f ",  A );
```

18)
```
float  A;
A=1.0 * 5/2;
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=1+a;
printf("%d   %d",  a,  b);
b=++a;
printf("%d   %d",  a,  b);
```

24)
```
A = 12;
printf("%d  %d",    A ,   -A );
A = -A;
printf("%d  %d",   A ,   -A );
```

25)
```
a=7%6;
printf("%d", a );
a=7%7;
printf("%d", a );
a=7%8;
printf("%d", a );
```

26)
```
printf("%c", 'A'+0);
printf("%c", 'A'+1);  // 'A'+1 → 'B'
printf("%c", 'A'+2);
printf("%c", 65);       // ASCII value of 'A' is 65
printf("%d", 'A' );
printf("%c", 66);
```

27)
```
a=234;
printf("%d ",  a%10 );
printf("%d ",  a%100 );
printf("%d ",  a%1000 );
```

28)
```
a=2345;
printf("%d",  a/10 );
printf("%d",  a/100 );
printf("%d",  a/1000 );
```

29)
```
a=2345;
a=a%10+a/1000;
printf("%d ", a );
```

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

31)
```
a=345;
printf("%d",   a%10 );
printf("%d",  (a/10)%10 );
printf("%d",   a/100 );
```

32)
```
a=345;
printf("%d",   a/100 );
printf("%d",  (a%100)/10 );
printf("%d",   a%10 );
```

41) The cost of each pen is rs:3/-, if we purchased N pens then what would be the cost. Now write a program to accept N as input and print total cost of N pens.

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

---

42) The cost of 5 pens is 12/-, if Ramu purchased N pens then what would be the approximate cost. Now write a program, where scan 'N' as no.of pens wanted by Ramu and show total cost of N pens.

input: N=20 ↲
output: cost is 48.00/- hint: cost is 12.0/5*N, here 12.0/5 gives each pen cost.

---

43) The cost of 5 pens is rs:12/- Ramu wanted to spend M rupees to purchase some pens then how many pens he can get for M rupees. Now write a program to scan M from Keyboard and show how many pens get.

input: M=48
output: Ramu can get 20pens for 48 rupees

4**4)** Ramu distributing N pens equally to 20 people, therefore each person gets same count of pens. After distributing N pens, he kept remaining pens with him. Now print how many pens distributed equally and how many kept with him. Here N is input (hint: use '/' and '%' operators)

calculations: N/20 → gives no.of pens equally distrusted to each person
N%20 → gives remaining pens left with him.

ip: if N is 103
op: Everyone gets 5 pens equally (N/20 → 103/20 → 5, so everyone gets 5)
Remaining pens with him is 3 (N%20 → 103%20 → 3, remaining pens with him is 3)

step1: scan input 'N' no.of pens
step2: print N/20 value. // this is like: printf("Everyone gets %d pens equally", N/20)
step3: print N%20 value. // this is like: printf("Remainning pens with him is %d", N%20);

---

45) The BigBite shop owner priced 3/- per each egg and also giving bonus eggs. He is giving one bonus egg for every 5 eggs purchase. For example,

if customer purchased 5 eggs then he get 1 egg as bonus
if customer purchased 10 eggs then he get 2 eggs as bonus
if customer purchased 15 eggs then he get 3 eggs as bonus
in this way calculate bonus eggs if he purchased N eggs.

Here the input of the program is number of eggs(N) wanted by customer and output is pay amount and number of eggs to be delivered to the customer.

input: eggs wanted by customer: 15 ↲
ouput: pay amount is 15*3 → rs 45/-
eggs delivered: 15+3 → 18 (here 3 is bonus eggs)

take variable names as: **N, payAmount, eggsToBeDelivered**

---

46) 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.

step4: print **N** as output.

---

47) Write a program to print multiplication table N for first 10 terms

ip: N=7

op: 7*1=7

7*2=14

7*10=70

printf("\n %d * %d = %d", N, 1, N*1);

printf("\n %d * %d = %d", N, 2, N*2);

---

48) Code to find Fahrenheit(F) temperature from a given Celsius(C)

ip: Celsius: 34

op: Fahrenheit: 93.2

procedure:

step1: Take variable names as: **C , F**

step2: Take **C , F** as **float** type, **float** is to store fraction values

step3: scan input value Celsius(**C**) from keyboard

step4: calculate Fahrenheit from Celsius, this is like **F**=(**C** × 9.0/5) + 32

step5: print Fahrenheit(**F**) as output

---

49) Write a program to calculate $5x^{10}+7x^2+9$.

ip: if '**x'** is 2 ip: if 'x' is 3

op: 5157 op: 59121

procedure:

step1**:** take variable names as: x,y. Here 'x' is to store input value and 'y' is to store output value

step2: take x,y as int-types

step3: scan input value into 'x'

step4: calculate output value as y=5*pow(x,10)+7*x*x+9

step5: print(y)

note: here pow() is a predefined library function to calculate power value.

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

---

50) Write a program to find value of $\frac{2x + 3}{2y + 3}$

ip: x is 1, y is 1 ip: x is 6, y is 1

op: 1 op: 3

procedure

step1: take variable names as: x , y , z.

step2: Take 'x','y' as int-type and 'z' as float-type

step3: here 'x','y' are input values from keyboard

step4**:** z=(2*x+3.0)/(2*y+3.0)

step5: print 'z' as: printf("result is %f", z);

---

51) Code to accept radius of circle from keyboard and print area and perimeter

ip: radius: 5

op: area=78.5, perimeter=31.4

calculations: Area of circle is πr^2, perimeter of circle is $2\pi r$

note: The symbol pie(π) or pie-value is not available or not known to C-language, so use the constant value 3.14 (22.0/7) in place of π symbol, for example: "area=3.14*radius*radius"

procedure:

step1: take variable names as: radius, area, perimeter (take all as float types)

step2: scan 'radius' as input

step3: find area // this is like: area=3.14*radius*radius

step4: find perimeter // this is like: perimeter=2*3.14*radius

step5: print area and perimeter like: printf("area is %f , perimeter is %f", area, perimeter);

---

52) 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>"

---

53) A shop keeper selling items with 12% discount, now our job is to scan item price from keyboard and print final price after discount.

ip: enter item price: 1500

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

procedure:

step1: take variable names as: item_price, discount, final_price

step2: scan item_price from K.B // KB→ Keyboard

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

step4: calculate final_price // final_price=price - discount

step5: print all (item_price, discount, final_price)

---

54) 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;

procedure:

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

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

step3: calculate HRA // HRA=24*basic**S**alary/100

step4: calcucate TA // TA=10*basic**S**alary/100

step5: find net**S**alary // net**S**alary=basic**S**alary+HRA+TA

step6: print net**S**alary

---

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

ip: 12 34

op: 34 12

Note: for swapping, one may write the code as Y=X and X=Y; here first when Y=X is executed then Y-value replaces with the X-value and we lose the current Y-value. So this code gives wrong output, following procedure gives an how to swap two values

procedure1: Let us say, one cup contained Milk, and another cup contained Water, now our job is to exchange Milk & Water in the cups. For this take extra empty cup, Now procedure for exchange is

step1: move milk-cup→empty-cup

step2: move water-cup→milk-cup

step3: move empty-cup→water-cup // at this moment empty-cup is not empty, it contained milk

In this way swap X & Y values in our program, here take extra variable (temp) for swapping

eg: temp=x; // this is like copying milk-cup to empty-cup

---

procedure2: (without using 'temp' variable)

subtract one with another value (X=X-Y), using this difference, we can swap X,Y values.

For example, Let X=10 , Y= 4

X=X-Y; // now X becomes 6

Y=X+Y; // now Y becomes 10, this is original value of X

X=Y-X; // now X becomes 4, this is original value of Y

This logic works for any values of X , Y. (but procedure1 is the best)

---

56) 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 // each hour has 3600 seconds, each minutes has 60 seconds.

---

57) Code to accept a time(N) in seconds format and print output in time format (H:M:S format)

ip: 7470

op: 02 : 04 : 30 // printf("%02d : %02d : %02d", h, m, s)

logic: divide **N** with 3600 and take the quotient as hours, (1hour=3600seconds, eg: H=N/3600)

divide **N** with 3600 and take the remainder(R). this R is the remaining seconds left after taking hours from N. for example R=N%3600; Again divide R with 60 and collect quotient and remainder. the quotient would be minutes and remainder would be seconds. (M=R/60; S=R%60)

---

58) 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,
   h3=h1+h2;
   m3=m1+m2;
   s3=s1+s2;
   // use following procedure
   1) convert two times into seconds.  (like above said problems)
   2) add two times which are in seconds now.
   3) distribute total seconds into H:M:S format.  (like above said problem)
   4) print output time.
}
```

---

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

ip: 4 5 ↲ (4+5i)
6 7 ↲ (6+7i)
op: 10+12i // print output as→ printf("%d + %d i", real3, img3);

let us take variables names: (real1, img1) + (real2, img2) → (real3, img3)

---

60) If 3 values entered through keyboard, write a program to find sum and average of them.

note: <u>take only two variables</u> in the program (let 'N', 'sum' are variable names)

ip: enter value1: 10
enter value2: 20
enter value3: 30
op: sum=60, average=20

process: read 3 values one by one into 'N' using three scanf() statements, after scanning every single value add it to 'sum', the Procedure is as follows:

step1. take two variable names as: **sum , N**
step2. read first input value into **N**
step3. store first **N** value into '**sum**'
step4. read second input value into **N** // of course, the second input value replaces the first input value of N.
step5. add second **N** value to '**sum**' // 'sum' already contained first value, so add this second N value to sum.
step6. read third input into **N**
step7. add this third **N** value to '**sum**'
step8. print(**sum, sum/3**) // printf("sum=%d, average=%d", **sum, sum/3**);

---

61) if two digit number like 47 is entered through 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, collect remainder & quotient.

If N=47 then the quotient will be 4 and remainder will be 7. This is as shown below,

```
10)  47  (4  →quotient
       40
    ----------
         7  → remainder

 sum =   N/10  +  N%10;
 sum = 47/10  + 47%10;
 sum =   4     +    7
```

---

62) If 'N' has 3 digits like 347, then print reverse of it (743)

Procedure:

step1: get first digit to 'd1' // check 21 and 22 problems for solution

step2: get second digit to 'd2'

step3: get third digit to 'd3'

Now generate reverse as: d3*100 + d2*10 + d1*1 → 3*100 + 4*10 + 7*1 → 300+40+7 → 347

---

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

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

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

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

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 1101 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,…

---

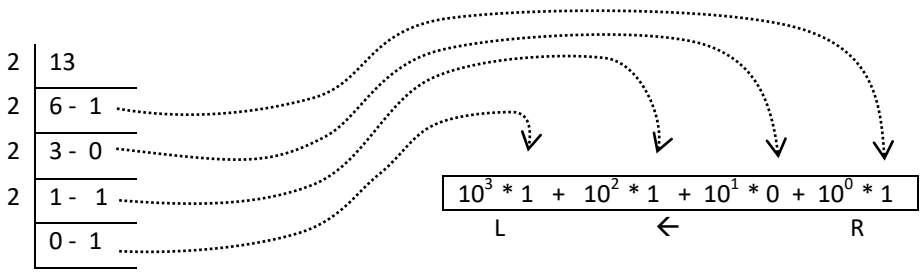
72) Write a program to find binary number from a given decimal number (Let the input is below 15)

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 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)

---

80) The BigBite shop owner priced 3/- per each egg and also giving bonus eggs. For every 100 eggs purchase customer get 2 extra eggs.

if he purchased 100 eggs then he get 2 eggs as bonus2.
if he purchased 230 eggs then he get 4 eggs as bonus2
if he purchased 502 eggs then he get 10 eggs as bonus2
in this way calculate bonus eggs if he purchased N eggs.

Here the input of the program is number of eggs(N) wanted by customer and output is pay amount and number of eggs to be delivered to the customer.

ip: 120 (N=120, no.of eggs wanted by customer)
op: total cost of 120 eggs is: 360rs/-
total deliver eggs count is : 120+2→122 (N + bonus)

take variables names as: N, pay, deliverEggs.

---

81) A fruit seller told 5 apples cost is 65rs/- and the customer wanted to buy apples for 100rs/-.
Here each apple costs 13rs/-, and he can get nearly 7 apples for 100rs/- and return change is 9rs/-
Now generalize this problem by writing a program, for example, if N apples cost is C, then what is the cost of each apple and how many apples customer can get for amount M, tell if any change to be returned.

List of variables names:

input: N → no.of apples, C → Cost of N apples, M→ Amount the customer can spent (buying amount).
output: eachAppleCost→ cost of each apples, countOfApplesForM → total no.of apples customer can get nearly for amount M, changeReturn → change to be returned to the customer.

ip: N=5, C=68rs, M=1000rs
op: each apple cost is 13.60rs
he can get 73 apples for 1000rs-
he get back 8rs/- change roughly.

note: use format string "%.2f" in printf() statement, for example printf("%.2f", 23.123456) → 23.12

if you use **type-casting/type-conversion** concept we can get result accurately.
Example for type-casting: (int)10.45 = 10, (float)14= 14.00, let a=45.67 then (int)a=45

---

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 ");
}
```

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( 0==A%2 )
{       printf("A is even ");
}
else
{       printf("A is odd");
}
```

ip: Let A=12
op: ?

ip: Let A=13
op: ?

7)

```
if( A==10 && B==20)
{       printf("Hello");
}
else
{       printf("World");
}
```

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

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

8)

```
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: ?

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

10)

```
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: ?

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.

```
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 (this is like starting space at paragraph).
This **indentation** makes the program easy to read and understand. It signifies which instruction is inside the if-statement and which is not.
Remember if we not followed indentation then code becomes difficult to understand.
C compiler does not force you to follow it, so it does not show any error if one ignored indentation.
(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");

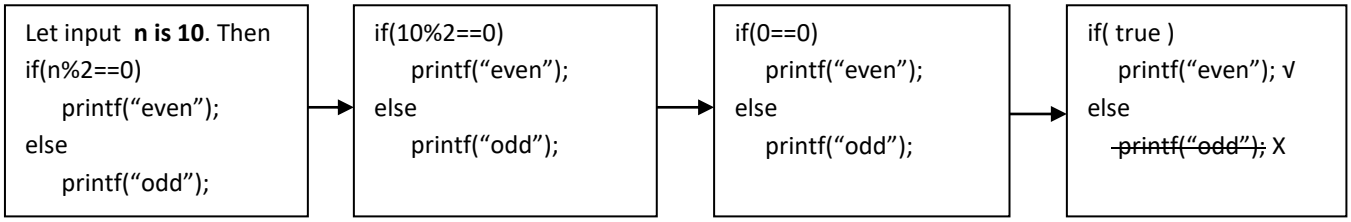
with indentation, no confusion
if(a==10)
{
  printf("red");
  if(b==20)
  printf("green");
  else
  { printf("blue");
  printf("white");
  }
  printf("black");
}
	Code without indentation, here lot of confusion
	if(a==10)
	{
	printf("red");
	if(b==20)
	printf("green");
	else
	{
	printf("blue");
	printf("white");
	}
	printf("black");
	}

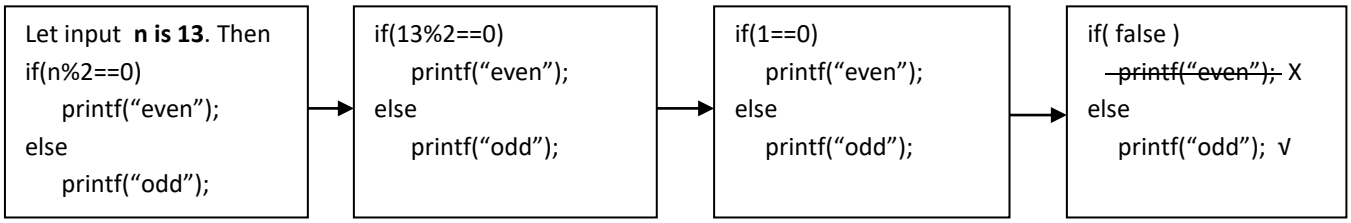
something wrong in this code, find it
if(A<B)
  printf("one");
  printf("two");
else
  printf("three");
  printf("four"); // not in if-body
printf("five");
	something wrong in this code, find it
	a=4, b=4;
	if(a=b)
	  printf("a, b are equal");
	else
	  printf("a,b are not equal");

==========

1) The operator % gives re-mainder of a division, for example 7%5 →2 , 12%2 → 0, (whereas 12/2 → 6)
We know, the numbers like 2, 4, 6, 8, etc are evens. If N is even then it divides perfectly with 2, it means the remainder of division should be equal to zero. In C program, the logic is

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





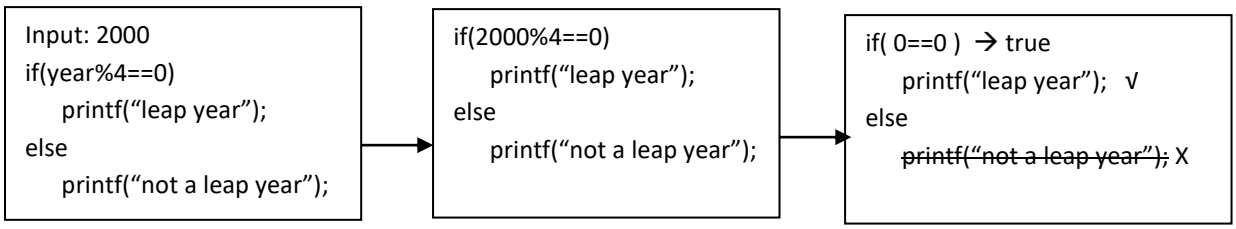
==========

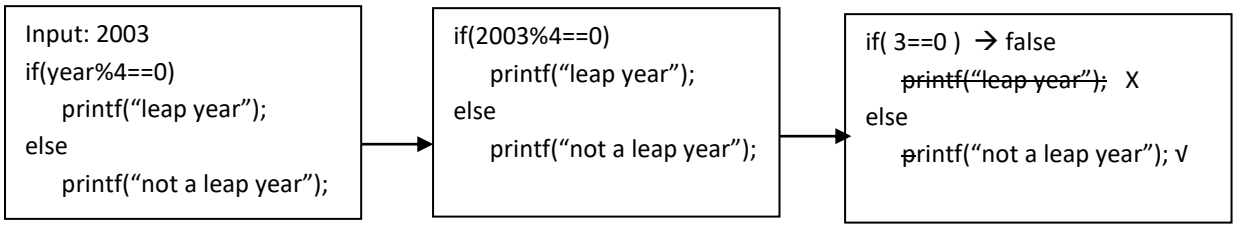
2) For every 4 years we get one leap-year, for example 2004, 2008, 2012 are leap-years.
If any year is divisible perfectly with 4, then it is said to be leap year otherwise not.
Perfectly divisible means, the remainder of division should be zero. year%4 → 0

ip: 2000 ip: 2005
op: 2000 is a leap year op: 2005 is not a leap year

```
if(year%4==0)
        printf("year is a leap year");
else
        printf("year is not a leap year");
```

The evaluation of **if-statement** is as follows

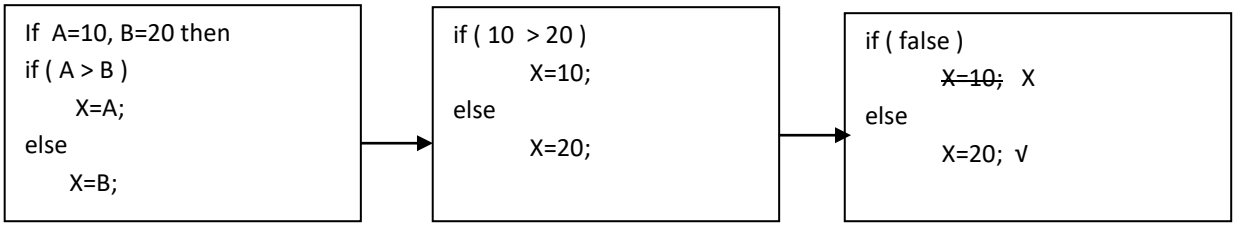




==========

3) Finding Biggest of two numbers, the code is as follows

```
if ( A > B )
     X=A;
else
     X=B;
printf(" Biggest is %d", X);
```



==========

4) If input is two numbers like A,B then print difference of A-B as output, and this output must be +VE.

ip: 10 30 ip: 30 10
op: 20 op: 20

```
void main()
{    int A,B,X;
     printf("enter two numbers :");
     scanf("%d%d", &A, &B);
     if ( A > B )
          X=A-B;
     else
          X=B-A;
     printf("difference output is %d", X);
}
```

About nested-if:

we know, constructing **"if in if"** is said to be nested-if. For example,

5)

```
if( age>20 )                          // outer-if
{
     if( weight<70)                   // inner-if 1
        printf("very good weight");
     else
        printf("overweight");
}
else
{
     if( weight<30)                   // inner-if 2
        printf("very good weight");
     else
        printf("overweight");
}
```

6) Finding given number N is –VE or +VE or ZERO

Following code explains how to write code using 3 simple if-statements and nested-if style.

```
if ( N<0 )
{       printf("N is –ve");
}

if ( N>0)
{       printf(" N is +ve");
}

if ( N==0)
{      printf("N is Zero");
}
```

→

```
// nested-if style  ( simple and best )

if ( N<0 )
        printf("N is –ve");
else
{
        if ( N>0)
             printf(" N is +ve");
        else
             printf("N is Zero");
}
```

==========

7) Code 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.

Writing code using 3 independent if-statements and nested-if

```
if( salary<=10000)
{       tax=0;
}

if( salary>10000  && salary<=20000)
{       tax=5*salary/100;
}

if( salary>20000)
{       tax=8*salary/100;
}
```

→

```
// nested-if style

if( salary<=10000 )
        tax=0;
else
{       if( salary>10000  && salary<=20000)
             tax=5*salary/100;
        else
             tax=8*salary/100;
}
```

8) Finding biggest of 3 numbers. Let A,B,C are 3 numbers

1) writing code using 3 independent if-statements 2) using nested-if style

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

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

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

→

```
// nested-if style

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

==========

9) Finding biggest of 4 numbers, extension to above program

```
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=D;
}
```



```
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;
        }
}
```

==========

10) Write a program to print age group of a person, this is as given below

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

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

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

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

```
if(age<13)
{       printf("child");
        weight=20;
}

if(age>12 && age<20)
{       printf("teenager");
        weight=40;
}

if(age>=20 && age<51)
{       printf("younger");
        weight=60;
}

if(age>=50)
{       printf("old");
        weight=80;
}
```



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

More about Logical operators && and ||

Sometimes nested-if can be eliminated using logical operators, these operators simplifies the code.

Observe the following code how logical operators simplifies the code

11) Following code finds given input number(N) is in b/w 10 to 20 or not?

```
if( N>10 )
{
      if( N<20)
            printf("yes");
      else
            printff("no, above 20");
}
else
      printf("no, below 10");
```

→

```
if( N>10 &&  N<20 )
      printf("yes");
else
      printff("no");
```

==========

12) Accepting 2 subject marks and printing result, Let subjects are A,B.
If student obtained <35 in any subject then student is "failed" or else "passed"

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

→

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

==========

13) extension to above program, taking 3 subject marks and printing result, Let subject are A,B,C.
If student obtained <35 in any subject then student is "failed" or else "passed"

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

→

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

14) Following code is opposite to above program logic

```
if(  A>=35)
{
        if( B>=35 )
        {
              if( C>=35 )
                    printf(“passed”);
              else
                    printf(“failed);
        }
        else
              printf(“failed”);
}
else
        printf(“failed”);
```

→

```
if( A>34  &&  B>34  &&  C> 34 )
      printf(“passed”);
else
      printf(“failed
```

15) Checking given date is valid or not

```
if( m==2)
{       if( y%4==0)
        {     if( d>29 )
                  printf(“invalid”);
              else
                  printf(“valid date”)
        }
}

if (m==2)
{       if( y%4!=0 )
        {     if( d>28 )
                  printf(“invalid”);
              else
                  printf(“valid date”)
        }
}

if( m==4)
{       if( d>30 )
              printf(“invalid”);
        else
              printf(“valid date”)
}
if( m==6)
{       if( d>30 )
              printf(“invalid”);
        else
              printf(“valid date”)
}
-----
-----
```

→

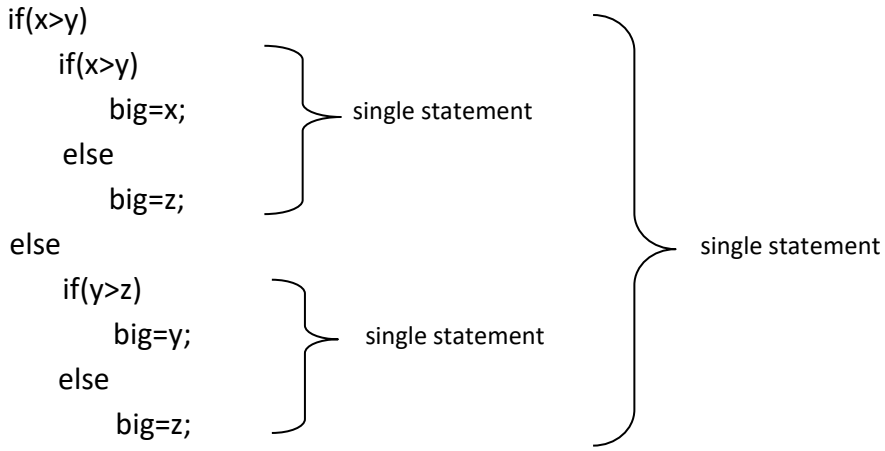
```
if(m==2 && y%4==0 &&  d>29 )
        printf(“invalid”);

if(m==2 && y%4!=0 &&  d>28 )
        printf(“invalid”);

if( ( m==4 || m==6 || m==9 ||m==11 ) && d>30 )
        printf(“invalid”);

-----
-----
-----
```

More about Pair of Braces{ }

Finding biggest of 3 numbers (without using logical operators)

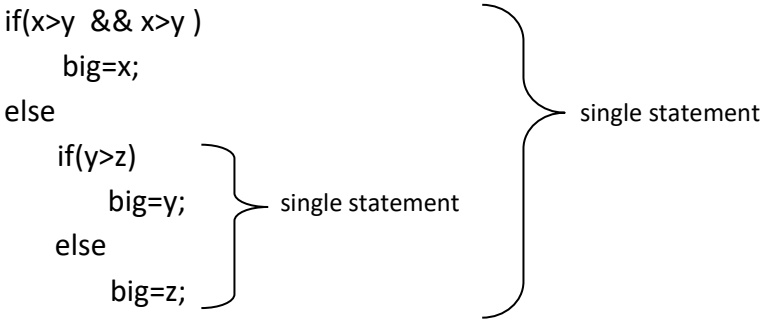
*** As per C language syntax, if-else control statement can be taken as single-compound-statement even though they seems to be two, for example, the following inner-if statements can be taken as single.**

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



In this way, in C, any total 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 2



Example2, after removing un-necessary braces

Normal nested-if with braces	After removing 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;
}
}
printf("\n biggest=%d", x) ;
	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

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 alternative exist.
So it is rearrangement of **nested-if** 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 automatically forms 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

Above program using if-else-if ladder style (finding biggest of 4 numbers)

Normal nested-if	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)
	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);

Above programs Nested-if style To if-else-if style

Finding given number N is –VE or +VE or ZERO

if (N<0)
printf("N is –ve");
else
{
if (N>0)
printf(" N is +ve");
else
printf("N is Zero");
}
	if (N<0)
	printf("N is –ve");
	else if (N>0)
	printf(" N is +ve");
	else
	printf("N is Zero");
	
	by removing unnecessary braces and writing all inner if-statements in the same line of else-part automatically forms a ladder-style

if(salary<=10000)
tax=0;
else
{ if(salary<=20000)
tax=5*salary/100;
else
tax=8*salary/100;
}
	if(salary<=10000)
	tax=0;
	else if(salary<=20000)
	tax=5*salary/100;
	else
	tax=8*salary/100;

Accepting 3 subject marks and printing result, Let subject are A,B,C.
If student obtained <35 in any subject then student is “failed” or else “passed”

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

→

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

==========

normal nested-if style to Ladder-style
if age<=12 then say child and weight is 20kg
if age>12 and age<=19 then say teenager and weight is 40kg
if age>19 and age<=50 then say younger and weight is 60kg
if age>50 then say old-aged and weight is 80kg

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

→

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

if-else

1) If integer, N is input through keyboard, write a program to print its absolute value [mod(N)]
the input number may be +ve/–ve entered by the user, but the output should be +ve.

logic: if input is –ve then convert into +ve by multiplying it with -1. // if N<0 then N is said to be -ve

ip: –12 ip: 14
op: 12 op: 14

procedure

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.

---

2) If two integers (A,B) are input through keyboard, write a program to find difference of them(A - B)
The output of difference must be printed in +ve, if result of A-B is –ve, then convert to +ve before printing

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

procedure1:

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

procedure2:

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
}
or 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

---

3) 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 value to X

}

or else

{ store B value to X

}

step4: now X is holding big value of (A,B), so print 'X' as biggest

---

4) If three integers(A, B, C) are input from KB, code to replace A, B, C 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

Procedure:

step1: take variable names as: A, B, C, temp; // temp 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.

---

5) If basic salary of employee scanned through 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

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 = basic salary + HRA + DA + TA]

---

6) Program to find given number(**N)** is an odd or even. If N is divisible perfectly with 2 then it is said to be even otherwise odd.

ip: 45 ip: 44
op: odd op: even

Procedure: If **N** is divisible perfectly with 2 means, the remainder of division must be equal to zero.
The modulus operator (%) gives remainder of division.

step1: Take N as int-type variable, and scan N from K.B
step2: rem=N%2; // N%2 gives remainder of division
step3: if rem==0 printf("N is even");
else printf("N is odd");

---

7) Any year is input through the keyboard, determine whether the year is leap-year or not.
method: if year is divisible perfectly with 4 then it is said to be "leap year" otherwise "not a leap year".

ip: 2005 ip: 2008
op: Non Leap year op: Leap year

---

8) Write a program to accept student marks(M) of one subject and print result pass/fail.
College adding 10 grace marks to all students, but after adding, it should not cross 100. (max marks is 100) if Ramu got 60 marks then his marks is 70, if he got 44 then his marks 55, if he got 93 then his marks 100. Now print pass/fail. if M<50 then print "fail" or else "pass"
procedure: scan M → add 10 to M → if M>100 then make M to 100 → now print "pass/fail"

---

9) for children, government providing nearly ticket free travelling for hill station by rope.
♦ the ticket cost for children <=16 years is 10/- 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
}
or else // age >16
{ here scan weight of a person
cost is weight/5*2
}
step4: print cost as output.

---

10) Accept a value**(N)** from keyboard and find whether it is +ve/–ve/zero.

ip: 12 ip: -12 ip: 0

op: +ve op: -ve op: zero

method1: try without using 'else' keyword (write 3 independent if-statements)

method2: try using normal nested-if style using 'else'

```
if ( N<0 )
      print  " N is -ve"

if ( N>0)
      print  " N is +ve"

if ( N==0)
      print  "N is Zero"
```

```
if ( N<0 )
      print  "N is –ve"
else
{     if ( N>0 )
            print  "N is  +ve"
      else
            print  "N is zero"
}
```

---

11) 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 (&&)

```
if (m1>=50  && m2>=50)   printf("passed")  else   printf("failed");
```

B) find using logical operator OR (||)

```
if (m1<50  ||  m2<50)   printf("failed")  else  printf("passed");
```

C) find without using logical operator (nested-if)

```
if(m1<50)
      printf("failed");
else
{     if(m2<50)
            printf("failed");
      else
            printf("passed");
}
```

```
if(m1>50)
{     if(m2>50)
            printf("passed");
      else
            printf("failed");
}
else
      printf("failed");
```

D) find without using logical operator and 'else' statement (use bool logic)

```
x=1;          // let us assume student has passed, so take 'x' as 1.
if(m1<50)   // if this condition is true, then student failed, so take 'x' as 0
     x=0;
if(m2<50)
     x=0;
// now result is in 'x' in the form of 1 or 0,  so check 'x' value and print "pass" or "fail"
if(x==1)  printf("passed");
if(x==0)  printf("failed");
```

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 and 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 problems(yes/no). We can use any other values instead of 1/0, but it is informal.

---

12) Rewrite above program by taking 3 subject marks.

1) using && operator
2) using || operator
3) using nested-if style (do not use logical operators here)

---

13) Code to find given input number(N) is in b/w 10 to 20 or not?

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

method1: using logical operator && (AND operator)
method2: using logical operator || (OR operator)
method3: without using logical operator(nested-if)
method4: try using 'bool' logic

---

14) if three integers are input through 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: "no, –ve is not exist"

method1: using logical operator || (OR operator)
method2: without using logical operator (nested-if)

---

15) Code 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/-

method1: **try without using 'else' keyword** (write 3 independent if-statements)
method2: try using normal nested-if style (this method is better than above)

---

16) if three integers(A,B,C) are input through keyboard, find biggest among them.

method1: try using normal nested-if style (as given below procedure)

step1: Take variable names as: A, B, C, X. // 'X' means e**x**tra variable, is to store output big value.
step2: if A>B and A>C then
{ above two conditions are true then big value is in A, so put A into X
}
or else
{ if control came to this else-part means, the big value is not in A.
the big value is in either B or C. so here compare B & C and store into X
}
step3: print biggest which is in X

---

17) if 4 integers are input through keyboard, find biggest among them. (Extension to above program)

method1: try using normal nested-if style
method2: try using ladder-style.

---

18) If marks of 2 subjects are input through keyboard, write a program to print result.

logic: Generally, to get pass mark, student must obtain >= 50 marks in 2 subjects, but university gave an exemption to the students. If he got 10 marks less in any one subject out of 2 then he is also passed. That is, he must get minimum 50 marks in one subject and 40 in other subject.

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

Logic: the possibility of pass marks can be found in following two cases.

case1: m1>=50 and m2>=40
case2: m1>=40 and m2>=50
case3: above two cases not satisfied then he is failed.

---

19) Above program checks 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

---

20) if three integers are input through keyboard, then find how many –ve values exist.

ip: -12 34 -42 ip: 52 64 -72 ip: 62 44 42
op: count=2 op: count=1 op: count=0

---

21) Code to accept 3 values from KB, here some values are +ve/-ve entered by the user, later 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:-5

---

22) If N is input through keyboard (0 to 99999), write a program to find how many digits exist.

ip: 234 ip: 3456 ip: 12234 ip: 3
op: 3 op: 4 op: 5 op: 1

method1: try using normal nested-if, for example,

```
if(N<10)
    count=1;
else
    if(N<100)
        count=2;
    ….
```

method2: Try also using ladder-style

---

23) Write a program to print age group of a person, this is as given below

if age<=12 then say child

if age>12 and age<=19 then say teenager

if age>19 and age<=35 then say younger

if age>35 and age<=60 then say middle-aged

if age>60 then say old-aged

method1: try using normal nested-if style

method2: try using if-else-if ladder style

---

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

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/-)

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.

Take variable names as: daysLate, fineAmount.

---

25) 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

method1:

if(y1>y2) then say date-1 is latest,

else if(y1<y2) then say date-2 is latest,

if above two conditions are false then years said to be equal, so we have to compare months and days.

method2: Compose date1 and date2 (3-values) into single value. (eg: k1=y1*10000+m1*100+d1)

Now compare k1 and k2 and find latest (this is simple than method1)

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

---

26) 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"

  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

---

27) 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

---

28) 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

---

29) 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

---

30) 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:

```
if( a+b+c != 180)
        printf("invalid input");
else  ……                    (don't print valid)
```

---

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

case1) equilateral
case2) isosceles with right angle
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

---

32) If date(month , year) is input through keyboard, write a program to print how many days exist in that month. (Let us say, the input date entered by the user is a valid-date)

- 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

---

33) If date(d,m,y) is input through 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>12) // if anyone is true then date is invalid, so replace bool with 0
bool=0;
else if(month is February + leap-year + d>29)
bool=0;
else if(month is February + non-leap-year + d>28)
bool=0;
else if(month is 4 or 6 or 9 or 11 and d>30)
bool=0;
step4: // now checking bool-value and print final result
if(bool is 1) then print("Given date is valid");
if(bool is 0) then print(" Given date is **in**valid");

---

34) 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

Home Work

40) 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)
                print(a,b,c);   // if true then → b is 2nd small
        else
                print(a,c,b)
}
…..
```

---

41) A number N which is b/w 10-100 input through 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

---

42) 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.

---

43) 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

---

44) 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)

---

45) The railway dept charges rs:1.20/- per every kilometer(km) travelled by passenger, and also giving discount. The input distance in 'km' travelled by passenger and output is fare (amount to be paid).
Note: take charge per km as fixed value [int charge=1.20]. That is, do not scan from keyboard.
The discount calculation is as follows

If km <= 100 then discount is 0/–

If km > 100 then discount is 50% on above 100 kilometers travelled.

For example,

- if km is 40 then: discount = 0 // here km<=100
- if km is 190 then: discount = charge*(190-100)*50/100; // 50% discount on above 100 km travelled.
- fare=km*charge – discount

ip: km=80 ip: km=190
op: fare is 96/- op: fare is 174/-

---

46) 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

---

47) 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

---

48) Code to accept 3 numbers for a,b,c, Let each number contained only single digit as input, now using a,b,c generate two numbers as X,Y. Where X is smallest, Y is biggest, later print difference of them(in +ve).

ip: a=3, b=1, c=5
op: X=135 , Y=531, difference is=396

---

49) 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

---

50) 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.

---

51) Find whether the given character is lower/upper case alphabet or digit or any other character?

ip: A ip: a ip: 9 ip: &

op: upper case alphabet op: lower case alphabet op: digit op: special symbol

Character	ASCII Values
A-Z	65-90
a-z	97-122
0-9	48-57

the sample code is:

```
char ch;
printf("enter character :");
scanf("%c", &ch):
if(ch>='A' && ch<='Z')     or   if(ch>=65 && ch<=90)
    printf("upper case alphabet");
else if(ch>='a' && ch<='z')   or   if(ch>=97 && ch<=122)
    printf("lower case alphabet");
else if(ch>='0' && ch<='9')   or   if(ch>=48 && ch<=57)
    printf("digit");
else  printf("other charater");
```

---

52) Write a program to scan an alphabet from keyboard and print its opposite case, if alphabet is not given then print such character as it is.

ip: A ip: a ip: 8

op: a op: A op: 8

The ASCII values for A-Z is 65-90, a-z is 97-122, so the difference is 32.

'A'+32→'a' and also 'A'+1 →'B'

'a'-32→'A' and also 'B'+1→'C'

```
if(ch>='A' && ch<='Z')   or   if(ch>=65 && ch<=90)
    ch=ch+32;
else if(ch>='a' && ch<='z')   or   if(ch>=97 && ch<=122)
    ch=ch-32;
```

---

while loop

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

2)
```
i=1;
while( i<=10 )
{   printf(" Hello ");
    i++;
}
```

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

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

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

6)
```
i=20;
while(i>0)
{   printf("%d ", i );
    i=i-2;
}
```

7)
```
i=1
while(i<=10)
{   printf("%d ", i );
}
i++;
```

8)
```
i=1;
while(i<=100000)
{   printf("%d ", i );
    i=i*10;
}
```

9)
```
i=5;
while( i<i+10 )
{   printf("%d ", i );
    i++;
}
```

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

11)
```
i=10;
while(i>0)
{   printf("%d  ", i );
    i--;
}
```

12)
```
i=1;
while(i>10)
{   printf("%d  ", i );
    i++;
}
```

13)
```
n=100;
while(n>0)
{   printf("%d  ", n );
    n=n/2;
}
```

14)
```
p=1;
while(p<100)
{   printf("%d  ", p );
    p=p*2;
}
```

15)
```
i=1;  j=10;
while(i<10)
{   printf("\n %d  and  %d  ", i , j );
    i++;
    j--;
}
```

16)
```
x=5;  y=7;   i=1;
while(i<10)
{   printf("\n  %d  and  %d", x , y);
    x=x+2;
    y=y+3;
    i++;
}
```

17)
```
i=1;
while(i<100)
{   if(i%2==1)
        printf("%d  ", i );
    i++;
}
```

18)
```
i=1;
while(i<20)
{   printf("%d  ", i );
    if( i%3==0 )
        printf("\n");
    i++;
}
```

19)
```
i=1;  N=15;
while(i<=N)
{  if( N%i==0 )
        printf("%d ",  i );
   i++;
}
```

20)
```
i=1;
while(i<11)
{  printf("%d ", i);
   i++;
}
i=10;
while(i>0)
{  printf("%d ", i);
   i--;
}
```

21)
```
i=1;   n=7;
while(i<=10)
{  printf("%d ", n*i );
   i++;
}
```

22)
```
n=7;    i=1;
while(i<=5)
{  p=n*i;
   printf("%d ", p );
   i++;
}
```

23)
```
i=1;n=7;
while(i<=5)
{  n=n*i;
   printf("%d ", n );
   i++;
}
```

24)
```
i=1; sum=0;
while( i<=4)
{  sum=sum+i*i;
   i++;
}
printf("%d ", sum);
```

25)
```
i=1;  sum=0;
while(i<=4)
{  sum=sum+i;
   printf("%d  ", sum);
   i++;
}
```

26)
```
i=1;  s=1;
while(i<6)
{  printf("%d ", s );
   s=s*-1;
   i++;
}
```

27)
```
int i=1;
while(i<6)
{  printf("%d  ", i );
   i=i*-1;
   i++;
}
```

28)
```
i=1;   s=1;
while(i<10)
{  printf("%d ",  s*i );
   s=s*-1;
   i++;
}
```

29)
```
p=1;
while(p<10000)
{  printf("\n%d ", p );
   p=p*10+1;
}
```

30)
```
i=1;   s=0;
while(i<6)
{  s=s+2;
   printf("%d ", s );
   i++;
}
```

31)
```
i=1;   s=1;
while(i<6)
{  s=s*2;
   i++;
}
printf("%d ", s );
```

32)
```
i=1;   f=1;
while(i<6)
{  printf("\n fact of %d  is  %d ", i , f );
   i++;
   f=f*i;
}
```

33)
```
i=3; s=0;
while(i<7)
{   s=s*10+i;
    i++;
}
printf("%d  ", s );
```

34)
```
int i=3; s=0; p=1;
while(i<7)
{   s=s+p*i;
    p=p*10;
    i++;
}
printf("%d  ", s );
```

35)
```
i=1;  x=2;
while(i<6)
{   printf("%d  ", x);
    x=x*x;
    i++;
}
```

36)
```
p=i=1;  x=2; y=3;
while(i<=y)
{   p=p*x;
    i++;
}
printf("%d  ", p);
```

37)
```
i=1;  sum=0; p=1;
while(i<=5)
{   sum=sum+p;
    p=p*2;
    i++;
}
printf("%d  ", sum );
```

38)
```
sum=0;  N=2345;
while(N>0)
{   sum=sum+N%10;
    N=N/10;
}
printf("\n  sum  is  %d", sum);
```

39)

'break' is a keyword, it stops the loop in the middle of execution The next example explains how it works. It prints output 1,2,3,4 and stops the loop.

```
i=0;
while( i<10 )
{   printf(" %d ", i);
    if( i==5)
        break;          // break throws the control out
    i++;                // so break stops the loop
}
```

40) The relational operator gives boolean value, for example

1<2 → true → 1

2<1 → false → 0

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

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

41) Note: in c, in condition place, the value zero is taken as false. Whereas other than zero is taken as true

→ the value 0 is false

→ the values 1, 0.5, -3, 10, -1.4 are true

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

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

while( -1.5 )     // always true, infinite loop
{   printf("world");
}
```

42)
```
N=5;
while(N)
{   printf(" %d ", N);
    N--;
}
```
op: ?

43)
```
N=1;
while( N!=5)
{   printf("%d  " , N);
    N++;
}
```
op: ?

Loops

write all following programs using while-loop but not with for-loop, it improves your logic skills quickly. Generally, when loop needs to be repeated certain number of times such as 10-times or N-times then for-loop is more recommended otherwise while loop is the choice. You can choose any loop as per your convenient, because the compiler generates same machine code for all types of loops. After solving all programs using while-loop, it is better to rewrite some programs in for-loop for practice. Remember, to solve these programs nested-looping is not required.

---

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 N-3 to N+3, here N is input value, take from keyboard.
ip: if N is 10
op: 7, 8, 9, 10, 11, 12, 13
Hint: in above program, the starting value is 10 and ending is 20. Here staring value is N-3 and ending is N+3

---

3) Code to print "Hello" for 10 times.
op: Hello Hello Hello Hello … 10 times

---

4) Code to print "+Hello" and "-Hello" for 13 times, this is as given below.
op: +Hello , -Hello , +Hello , … 10 times

```
logic:  if( i%2==1)  printf("+Hello");        // when 'i' is odd
        else  printf("-Hello");               // when 'i' is even
```

---

5) Accept N from KB, if N is -ve then print "Hello" for 10 times, if N is +ve then print "World" for 10 times.
ip: N is -12 (-ve) ip: N is 12 (+ve)
op: Hello, Hello, Hello … 10 times op: World, World, World … 10 times

---

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

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

---

8) 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

---

9) 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

---

10) Code to print 1, 2, 4, 8, 16, 32 ……for N times, where N is input. For example,
ip: N=5
op: 1 2 4 8 16 (N terms, where N is 5)
take: repeat loop N times using 'i' ; here 'i' value is 1,2,3,4,5,….N
take 'p' to generate and print 1,2,4,8,16, ….etc. Here double the p value for next cycle.

---

11) 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

Hint: Look at program-2

---

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) 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

method1: take loop variable 'i' with 1, and increment it by 2 to get next odd value in the loop.

method2: take loop variable 'i' and increment it by 1 in every cycle, but print 'i' value when it is odd.

for example:
```
if(i%2==1)   // write this code inside while-loop for odd checking
{   printf("%d ", i);
}
```

---

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

logic: write printf() statement as printf("%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

---

15) 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");

---

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

ip: Let N is 7

op: 7, 14, 21, 28, 35, 42, …. up to 10 times

---

17) 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);

---

18) 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;

---

19) 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

---

20) If two input values 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 entered 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.

---

21) 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.

Here we need to write two loops but need to be executed only one, which is based on L&U values.

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) { ... }
```

---

22) Code to accept numbers one by one until last input value is zero, when zero is entered then stop scanning and print sum of all values.

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

logic: here the scanf() statement need to be written inside while loop, because we need to scan continuously until last input is zero. Here take while loop as infinite loop while(1) { … } and also use 'break' statement to stop the loop when input is zero. Here the value '1' as loop condition represents infinite loop, off course the loop stops with 'break'. The partial code is as follows

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

---

23) In a school, every cricket player student has to play 10 balls, and school is offering gold coin prize for those who scored >= 20 runs, and also extra gold coin if he blast at least one 6 shot.

Now write a code to scan 10 values one by one as input, these values are nothing but runs made in each ball. Here sum all runs scored by a student, later show whether student gained cold coin or not?

note: if input is -1 then player is out in the middle of 10 balls then stop scanning

input1: 2 (2 runs made at first ball
input2: 0 (0 runs made at second ball)
input3: 4 (boundary shot)
input4: 1 (single run)
input5: 6 (six shot)
….
input10: 2 (2 runs)
op: total runs made 25 with six shot, prize is 2 gold coins.

input1: 2 (2 runs at first ball)
input2: 0 (0 runs at second ball)
input3: 6 (6 runs at third ball)
input4: -1 (player out)
op: total runs are 8, lost the game

---

24) 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 in time 0 to 23 :");
            scanf("%d", &H);
            if(H<0 || H>23)
                -----
        }
```

---

25) 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.

---

26) 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

---

27) 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 value: 3 5
op: output = 15

---

28) 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 "yes" or "no")
The logic like: sum=sum+i*i, where i=1 to 10 do

---

29) Program to find sum of 1+2+3+4+5+ ……+N
note: to find sum of 1+2+3+4+…+N, do not use formula like N*(N+1)/2
ip: N = 5
op: sum=15

---

30) Program to find sum of odd numbers 1+3+5+7+9+ … N terms. (the N^{th} term is 2*N-1 in odd series)
ip: N=5
op: 25 (1+3+5+7+9)

---

31) Code to find product of 2*2*2*2*2….N times. Here N is input value.
note: do not use pow() function
ip: enter N value:5
op: output = 32

---

32) Write a program to find factorial of N.
ip: let N is 5
op: 125 (5*4*3*2*1, we can also calculate this value as 1*2*3*4*5)

---

33) Program to find sum & product of 1 to N [1+2+3+....+N, 1*2*3*....*N]
ip: N = 5
op: sum=15
product=120

---

34) 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

---

35) 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 (increment 'i' every time by 1)
take the variable 'p' to produce and print 1, 2, 4, 8, 16, 32,… (multiply 'p' with 2 to get next value)

---

36) 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
logic: 1. take 'i' for looping, where 'i' is 1, 2, 3, 4, 5, 6, ….N
2. take 'p' to generate 1, 2, 4, 8, 16, 32, ….. (2^0, 2^1, 2^2, 2^3, 2^4, 2^5)
3. take 'sum' and add 'p' values to sum.

---

37) 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)

---

38) 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

---

39) Program to print 7, -7, 7,-7, 7, -7, …N times
logic: take 'i' for 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.
the sign of 'V' alternatively changes to +ve to -ve and –ve to +ve.

---

40) 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"

---

41) Program to print 1, -2, 3, -4, 5, -6, 7, …N times
logic: 1) take a variable 's' and 'i'
the variable 'i' for looping from 1 to N
the variable 's' for sign, its sign changes alternatively to +1, –1, +1, –1, +1, –1, …etc.
2) print 's*i' as output, this is like printf("%d ", s*i);
3) increment 'i' by 1
4) change 's' to opposite sign
5) repeat step2, step3, step4 for N times
In the loop, the change of 'i' and 's' values as given below
'i' → 1, 2, 3, 4, 5, 6, …etc
s → +1, –1, +1, –1, +1, –1, …et
s*i → 1, –2, +3, –4, +5, –6, …etc

---

42) Program to print the value of each term 1/1, 1/2, 1/3, 1/4, …N times. // printf("%0.2f ", 1.0/i)
output as: 1 0.5 .33 .25 0.2 0.16 ….

---

43) Program to find sum of 1/1 + 1/2 + 1/3 + 1/4, …N times.
ip: N = 5
op: 2.28

---

44) Program to print value of each term 1/2, 2/3, 3/4, 4/5 …N times // print(i/(i+1))
output as : 0.5 .66 .75 0.8 ….

---

45) find sum of 1/2 + 2/3 + 3/4 +.....+ N/(N+1) [0.5 + 0.66 + 0.75 + 0.8 + 0.83 + …N times]
ip: n=5
op: sum=3.55 [0.5 + 0.66 + 0.75 + 0.8 + 0.83 → 3.55]

---

46) Program to print value of each term 1/2, 3/4, 5/6, 7/8, … N times // printf("%.02f", i/(i+1))
output: 0.5 0.75 0.83 0.87 ….

---

47) 2/9 - 5/13 + 8/17 - 11/21 …. N times [0.2222 + -0.3846 + 0.4705 + -0.5238 + 0.5600 + …. N times]
ip: N=5
op: sum=0.34 43 [0.2222 + -0.3846 + 0.4705 + -0.5238 + 0.5600 → 0.3443]
Hint: sum = sum + sign*x/y for N times
where 'sign' changes alternatively +1, -1, +1, -1, +1, -1, …etc
'X' starting value is 2 and increment every time by 3,
'Y' starting value is 9 and increment every time by 4

---

48) 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

---

49) 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 sum values in each cycle is 1, 1+2, 1+2+3, 1+2+3+4, 1+2+3+4+5,etc

By adding all these 'sum' to variables to another variable(say 'sum2') then we get final sum.

---

50) 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)

---

51) 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

---

52) 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.

---

53) $X^1/1!$ + $X^2/2!$ + $X^3/3!$..... 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]

---

54) $X^1/1!$ - $X^3/3!$ + $X^5/5!$ - $X^7/7!$ ..... 5 times. [sine series]

ip: x=3, N=5

op: sum=0.1453 [(3.0) + (-4.5) + (2.025) + (-0.4339) + (0.05424)]

---

55) 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. the loop repeats as given below

if(N%1==0) then '1' is factor of N

if(N%2==0) then '2' is factor of N

if(N%3==0) then '3' is factor of N → in this way check with all possibilities from 1 to N.

ip: enter **N** value:18

op: 1, 2, 3, 6, 9, 18

Count of factors=6

---

56) 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.

---

57) If N is scanned through the keyboard, write a program to print big factor **other than N**.

hint: Generally, for any number, the possible factors lies in range 1,2,3,…N/2, ~~N/2+1, N/2+2……N-1~~,N.

That is , there should not be factors after N/2 except N.

for example, if we take 100 then possible factors are 1,2,3,4,5,…48,49,50,~~51,52,53,…98,99~~,100.

The value 100 never divides with 51, 52, 53,…97, 98, 99. So it is useless to check with these numbers.

Our requirement is to find big factor other than N, so it is wise to check from N/2 to 1.

logic: divide N with N/2 to 1, that is check with all possibilities from big to small (from N/2 to 1), which ever divides first then it is big factor.

---

58) 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 input two values into x , y

step2: find smallest of x , y // because common factors exist upto small of x,y

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

step3: repeat loop from 1 to 'small'

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

---

59) 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 (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
```

step5: stop

---

60) Program to accept a number **N** and find whether it is perfect or not?

perfect: if sum of all factors is equal to given N then it is said to be perfect. (don't take N itself as a factor)

logic: Check for factors from 1 to N/2 and add all divisible to 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

---

61) In examinations, interviews, viva,…etc there always one program being asked, that is, none other than "prime number logic". Write a program to find the given number N is prime or not.

Prime numbers divide only with 1 & itself (ie., they do not divide with any other numbers except 1 & N)

ip: n = 17 ip: n = 18

op: yes, it is prime op: no, it is not prime

logic1: As we discussed earlier, for any number, factors lie between 2 to N/2 (by excluding 1 & itself). There should not be factors after N/2, so it is wise to check prime-ness from 2 to N/2 instead of all. During checking process, if N is divided then stop the loop and say "not prime". If not divided till end then say "prime" A beginner write prime number logic in wrong way as

```
i=2;
while( i <= N/2 )
{   if(N%i==0)  printf("\n not prime");
    else  printf("\n 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

```
i=2;   bool=1;          // assume N is prime, so take bool as 1 (true).
while(i<=N/2)
{   if(N%i==0)
    {   bool=0;     // here N divided, therefore N is not prime, so set bool to 0 and stop the loop
        break;
    }
    i++;            // if N is not divided then check with next 'i' value
}
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)

---

62) 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 digit by digit from N and add to 'sum', for this divide N continuously with 10 and collect the remainders, the remainders are nothing but digit after digit from last to first in N, and add this digits to sum.

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: add this remainder to 'sum'

sum=sum+R (adding to sum)

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 until N>0

---

63) 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("\n5 is exist ");
        else
                printf("\n5 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 <u>shows wrong output as</u> (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.

---

64) 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, sumOfEvens, sumOfOdds

---

65) If a number N is input through the keyboard, write a program to find whether it is Armstrong or not?

logic: If sum of cubes of all digits of N is equal to the N itself, then it is called Armstrong number.

eg: 153 → (1*1*1)+(5*5*5)+(3*3*3) → 153

ip: 153 ip: 445

op: "yes, the number is an Armstrong" op: "no, the number is not an Armstrong"

---

66) Write a program to find big & small digit of a given number.

ip: 2715

op: big=7 , small=1

logic: take variable called 'big' with value zero, now compare each digit(D) with 'big' , if big<D then take D into 'big', finally 'big' contains biggest value. Likewise find 'small' also.

---

67) Write a program to find reverse of given number

ip: 2345

op: 5432

step1: Let N is input number, take REV to store reverse value. Initially set REV to zero.

step2: get last digit(D) from N and insert it into REV by doing REV=REV*10+D

step3: to get next digit from N, now remove current last digit from N by doing N=N/10

step4: repeat step2, step3 until N>0

REV = REV * 10 + n%10 (here D=N%10)

= 0 * 10 + 5 → 5

= 5 * 10 + 4 → 54

= 54 * 10 + 3 → 543

= 543 * 10 + 2 → 5432

= 5432

---

68) 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: 1234

op: palin-drome. op: not palin-drome

logic: After finding reverse of a given number like above said, the value of N becomes 0, because in the loop the instruction N=N/10 makes the N value to zero. So before looping, store N value into some variable like 'temp', after finding reverse of N, compare reverse value(REV) with 'temp' to check palindrome.

---

69) 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

---

70) Write a program to find given number is valid binary or not?

ip: 1101 ip: 1201

op: yes, valid binary op: no, not valid binary

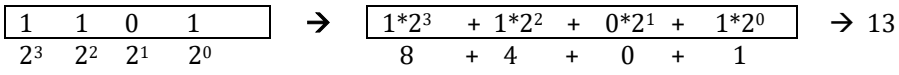
logic: extract digit by digit from N, if any digit>1 then stop the loop and say "it is not valid binary".
or else say 'it is valid binary'.

---

71) Write a program to find decimal number from a given binary number

ip: 1101

op: 13



step1: multiply all digits of N with 2^0, 2^1, 2^2, 2^3, 2^4… from right-to-left to get values of 2^0, 2^1, 2^2, 2^3…, use 'p' and multiply it with 2 .

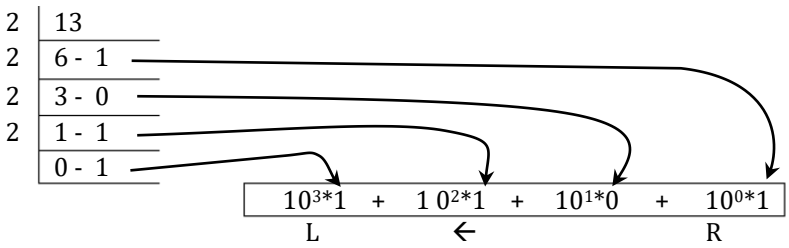
step3: The sum of all such products forms a decimal number.

---

72) Write a program to find binary number of a given decimal number

ip: 13

op: 1101



logic: divide continuously N with 2, and collect remainders(R), after getting each R, multiply with $\mathbf{10^i}$ and add to 'sum' variable. [here i=0,1,2,3,4,5,…]. Do not use pow() fn.

---

73) Write a program to print Fibonacci series up to 10 terms.

process: The first two terms in this series are 0, 1 and remaining values generated by adding previous two values: 0 1 1 2 3 5 8.....

step1: Let us take first two terms are x=0, y=1;

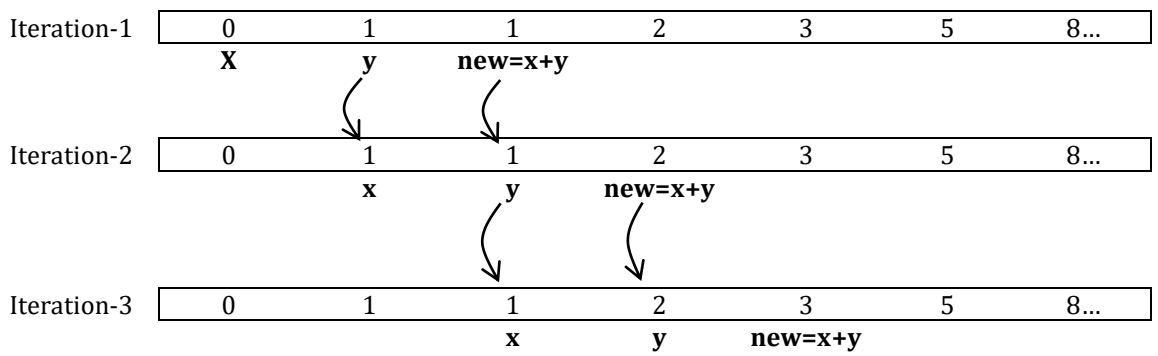
step2: Print the term 'x'.

step3: Generate next term by adding **'x+y'** to '**new'**

step4: Now shift '**x'** to '**y'** and '**y'** to **'new'** for next cycle (for this do X=Y, Y=new)

step5: Repeat this process for 'N' times

Let us see how the x, y are advancing in every iteration of loop



---

74) Write a program to find GCD of two numbers. (GCD/HCF → Greatest Common Divisor)

ip: 12, 18

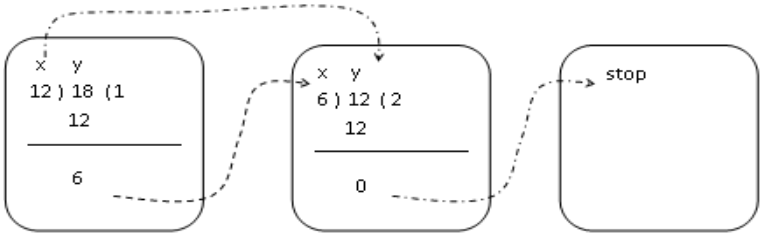
op: 6

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.



Home Work

80) 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.

---

81) 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

---

82) 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.

---

83) 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

---

84) If two input values taken from keyboard as lower and upper limits, write a program to print odd numbers between them. Sometimes the input values may entered in reverse order, for example 23, 14 (lower>upper) anyway print them in ascending order.

ip: 15 32
op: 15 17 19 21 23 25 27 29 31

ip: 14 31
op: 15 17 19 21 23 25 27 29 31

ip: 30 14
op: 15 17 19 21 23 25 27 29

---

85) If any value **N** taken from keyboard, write a program to print odd numbers from **N** to 1.
The input value **N** may be odd/even entered by user.

ip: enter N value:15
op: 15, 13, 11, 9, 7, 5, 3, 1

ip: enter N value:16
op: 15, 13, 11, 9, 7, 5, 3, 1

method1: 1) check N value, if N is even then do N=N-1 (this is to change N from even to odd).
2) now take loop and print odds from N to 1 by decrementing N by 2 every time.
method2: decrement N-- upto 0 using loop, but print N when it is odd. eg: if(N%2==1) printf("%d ", N).

---

86) Code to accept numbers one by one from keyboard until last input value is 0, when zero is entered then stop scanning, later print count of -ve numbers.

ip: enter value1: 11 ↲
enter value2: -3 ↲
enter value3: 44 ↲
enter value4: 55 ↲
enter value5: -6 ↲
enter value7: 0 ↲ [0 is end of input]
op: -ve count = 2

logic: 1) Use variable 'c' to count '–ve' numbers
2) Increment 'c' when input(N) < 0

---

87) One monthly creditor lends money to the customer and he wants to repay within one month, if customer failed to repay back then he adds interest to the principle for next month. This is simple interest for month wise but if customer failed for longer period then it goes like compound interest.
Any way show how simple interest is accumulated in every month.

for simple interest(si), the formula is : si = p*t*r/100, p→principle, t→time, r→rate of interest
Write a program to accept only principle (p) from keyboard and show how simple interest(si) is accumulated in every month. (Show for 5 months). Let us take interest rate as 2/- and time as 1 month (do not scan r & t)
ip: principle=100000
op: after month-1 , principle=100000, interest=2000, repayment(p+si)=102000
after month-2 , principle=102000, interest=2040, repayment=104040
after month-3 , principle=104040, interest=2081, repayment=106121
after month-4 , principle=106120, interest=2122, repayment=108243 …
logic: take while loop and repeat for 5 times

---

88) 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	**if N=15 then**
while(1*1<16) is true	while(1*1<15) is true
while(2*2<16) is true	while(2*2<15) is true
while(3*3<16) is true	while(3*3<15) is true
while(4*4<16) is false	while(4*4<15) is false
now 4*4==16 is true, so print "yes"	now 4*4==15 is false, so print "no"

8**9)** 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 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 ")

---

90) program to print how many ways the input 'N' can be written in multiples.

ip: N=100
op: 100 * 1 =100
50 * 2 = 100
25 * 4 = 100
20 * 5 = 100
10 * 10 = 100

method: Check for factors of N from 1, 2, 3, 4, 5,… until i<=N/i, and print output as above shown.
use printf() statement as printf("\n %d * %d = %d", n/i, i, n);

---

91) Code to find sum of even & odd positions digits in a given number (right to left). Let N is: 789453

← 7	8	9	4	5	3 ←
6th place (even place)	5th place (odd place)	4th place (even place)	3rd place (odd place)	2nd place (even place)	1st place (odd place)

ip: 789453

op: even place digits sum = 7 (5+9+7), odd place digits sum = 9 (3+4+8)

logic: **step1:** get the last digit from N and add to 's1' (this will be the odd place digit)

step2: remove last digit from N (**N=N/10)**

step3: get the next last digit from N and add to 's2' (this will be the even place digit)

step4: remove last digit from N **(N=N/10)**

repeat above steps until N>0

---

92) Write a program to count number of digits in a given number

ip: 29431

op: count=5

---

93) 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.

---

94) 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)

---

95) Write a program to print <u>right most odd digit</u> in a given number, if odd digit not exist then print the message "odd not found".

ip: 234**5**6 ip: 2468

op: 5 op: odd digit not found

---

96) Write a program to print <u>left most odd digit</u> in a given number, if no odd digit exist then print message "no odd digit found".

ip: 2**3**451 ip: 2468

op: 3 op: odd digit not found

---

97) Write a program to find second big digit in a given number.

ip: 2751 ip: 5555

op: 5 op: 5

---

98) Write a program to print given number N in English words.

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

the code to generate 'd' value as

```
d=1;
while(N/d>9)
{   d=d*10;
}
```

step2: now take one more loop, 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 [if N=2345, q=2345/1000, then 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

---

99) Write a program to subtract '1' from all digits. If input is: 4056, then output is: 2945

step1: take 'p' and generate its value based on input N. for example,

if N=123 then generate 'p' to 111 [here N has 3 digits]

if N=1234 then generate 'p' to 1111 [here N has 4 digits]

step2: now subtract 'p' from N [4056-1111 → 2945]

step3: print N value.

---

100) Write a program to accept a number and print after swapping first and last digit of a number.

ip: 2345

op: 5342

---

101) Program to print last Fibonacci number which is fall below N. Here N input.

ip: 12 ip: 8 ip: 13 ip: 35

op: 8 op: 8 op: 13 op: 34

---

102) 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~~

---

103) 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

---

104) 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;
}
```

---

105) Write a program to find LCM of 3 numbers

ip: 12 18 45

op: 180

1. Let x, y, z are three input numbers
2. Start finding LCM of three with first factor 2
3. Now divide each x, y, z with 2, and decrement all divisible numbers to quotient obtained in the division.
   For example, if x is divided with 2 then decrement x=x/2
4. If any x, y, z is divided in step3 then take 2 as factor in LCM. (LCM=LCM*2)
5. Now repeat step-3, step-4 as long as 2 divide any x, y, z
6. if 2 no longer divided, then next try with next factors 3, 4, 5…etc,
   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

---

106) 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

---

107) 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 ;
}
```

---

108) 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.

---

109) Write a program to accept date from keyboard and check whether it is valid or not, if not then scan again & again till user entered a valid date, finally print the date.

ip: 31-2-2001 ip: 21-2-2001

op: invalid date, try again op: yes, valid date

---

110) Write a program to accept a valid date from keyboard and increment it by N days.

ip: 1-1-2009 and 366

op: 2-1-2010

logic: step1: take loop for N times

step2: increment date by 1-day in every cycle of loop.

step3: after looping, print date.

---

111) 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

---

112) Accept two-dates from keyboard and print their difference in days, let the two dates are valid.

Algorithm:

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
    d1++;
    -----
    -----
}
printf("difference count is = %d", count);
```

ip: date1 = 1-1-2009

date2 = 2-1-2010

op: difference count is = 366

---

113) 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.

---

114) Write a program to accept month and year from keyboard and print calendar of that month.

logic: Using previous program we can solve easily.

Nested Loops

1)
```
i=1;
while( i <= 5 )
{     printf(" A ");
      j=1;
      while( j <= 3 )
      {     printf(" B ");
            j++;
      }
      printf(" C ");
      printf(" \n ");          // goto next line
      i++;
}
```

2)
```
i=1;
while( i <= 5 )
{     printf(" A ");
      j=1;
      while( j <= 3 )
      {     printf(" %d  ",  j );
            j++;
      }
      i++;
      printf(" B \n");
}
```

3)
```
i=1;
while( i<6 )
{     printf("start ");
      j=1;
      while( j<4 )
      {     printf("%d  ", i );
            j++;
      }
      i++;
      printf(" end\n");
}
```

4)
```
i=1;
while( i<6 )
{     j=3;
      while( j<8 )
      {     printf(" %d  ", j );
            j++;
      }
      i++;
      printf( "\n" );
}
```

5)
```
i=1;
while( i<4 )
{     j=1
      while( j<6 )
      {     printf("\n %d :: %d", i, j );
            j++;
      }
      i++;
      printf("\n");
}
```

6)
```
i=1;
while( i<=8 )
{     printf("\n table %d :: ", i);
      j=1;
      while(  j<=10 )
      {     printf("%d  ",  i*j );
            j++;
      }
      i++;
}
```

7)
```
i=8;
while( i>=1 )
{     j=1;
      while(  j<=i )
      {     printf("%d  ",  j );
            j++;
      }
      printf("\n");
      i--;
}
```

8)
```
i=1;
while( i<=8 )
{     j=i;
      while( j<=8 )
      {     printf("%d  ",  j );
            j++;
      }
      printf("\n");
      i++;
}
```

9)
```
i=4;
while( i<=10 )
{       j=1;
        while( j<=i )
        {     printf("%d  ", j );
              j++;
        }
        i++;
        printf("\n");
}
```

10)
```
i=1;
while( i<=8 )
{    j=1;
     while( j<=5 )
     {     printf("%d  ", i );
           j++;
     }
     i++;
     printf("\n");
}
```

11)
```
i=9;
while( i>0 )
{     j=i;
      while( j<=9 )
      {    printf("%d  ", j );
           j++;
      }
      i--;
      printf("\n");
}
```

12)
```
i=9;
while(i>0)
{     j=9;
      while(j>=i)
      {     printf("%d  ", j );
            j--;
      }
      i--;
      printf("\n");
}
```

13)
```
for(k=i=1; i<=5; i++)
 {     for(j=1; j<=5; j++)
            printf("%d", k++);
       printf("\n");
 }
```

14)
```
for(i=1; i<6; i++)
 {    for(j=1; j<=4; j++)
           printf("*");
      printf("\n");
 }
```

15)
```
n=2835;
while(n>0)
{     r=n%10;
      for(j=1; j<=r; j++)
           printf("*");
      printf("\n");
      n=n/10;
}
```

16)
```
for(i=1; i<6; i++)
 {    for(j=0; j<5; j++)
           printf("%d", 'A'+j  );
      printf("\n");
 }
```

17)
```
for(i=1; i<6; i++)
 {    for(j=1; j<=5; j++)
           printf("%d",  j  );

      for(j=5; j>=1; j--)
           printf("%d",  j  );

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

18)
```
for(i=1; i<6; i++)
 {    for(j=1; j<=i; j++)
           printf("%d",  j  );

      for(j=i; j>=1; j--)
           printf("%d",  j  );

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

19)
```
for(i=1; i<40; i++)
 {
      printf("%d",  i  );

      if( i%5==0)
           printf("\n");
 }
```

Nested Loops

Solving the following patterns makes the programmer command over the nested loops, thereafter we can easily handle complex data like 2D arrays such as matrices, strings, files, calendar, networks routing for shortest path, etc.

1) produce the following output pattern

```
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
```
➔ ➔

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 367 to 7654
op: 77, 88, 99, 101, 111, 121, 131, ….

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++)
{       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

Home Work

40) Write a program to print previous prime of a given number N (N>2). (N can be prime | non-prime)

ip: 10 ip: 100 ip: 19
op: 7 op: 97 op: 17

Note: start finding prime from N-1 to 2 (in reverse order)

41) Write a program to print next prime of a given number N. (N can be prime | non-prime)

ip: 13 ip: 100 ip: 17
op: 17 op: 101 op: 19

Note: start finding prime from N+1 to infinate(in reverse order)

42) 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)

43) 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.

44) 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.

45) Write a program to generate all combinations of 1, 2, 3 using three nested loops.

Output:

```
123
132
213
231
321
312
```

46)

```
1
01
010
1010
10101
----------
8 rows
```

47)

```
 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
```

48)

```
ip: 5263
op: *****
    **
    ******
    ***
```

49) 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
```

50) ip: n=5

```
1 1 1 1 1
2       2
3       3
4       4
5 5 5 5 5
```

51)

```
    1
   222
  33333
 4444444
555555555
```

52)

```
555555555
 4444444
  33333
   222
    1
```

53)

```
555555555
 4444444
  33333
   222
    1
   222
  33333
 4444444
555555555
```

54)

```
    1
   222
  33333
 4444444
555555555
 4444444
  33333
   222
    1
```

1D-Arrays

---

1) 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 5

op: yes, -ve exist op: no, -ve not exist

Logic: if any value a[i]<0 then –ve exist, or else not.

```
void main()
{   int a[5], …..;
    printf(“enter 5 values:”);
    for( i=0;  i<5; i++)
    {   scanf(“%d”, &a[i] ) ;
    }
    ----
```

---

2) 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.

---

3) Code to search an element ‘11’ is exist or not? Let array contained 5 values scanned from KB.

ip: 4 6 11 12 5 ip: 4 16 13 12 5

op: 11 exist op: 11 not exist

---

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 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 %d :”, i+1);
        scanf(“%d”, &a[i] ) ;
    }
    ----
}
```

```
running of program
input
~~~~~~~~~~~~~
enter how many input values: 6
enter value 1: 7
enter value 2: 10
enter value 3: 14
enter value 4: 23
enter value 5: 16
enter value 6: 18
output
~~~~~~~~~~~~~
sum is : 88
average is : 14.66
big is : 23
```

7) 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

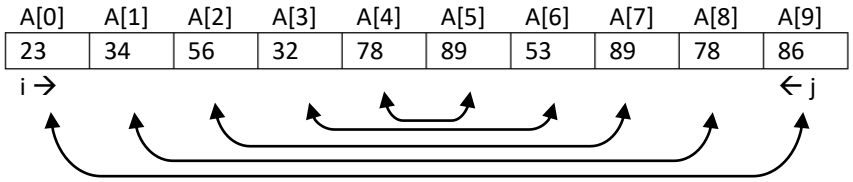
note: if 3 pair of elements found in adjacent place then take them as one pair(not two pairs), for example 8.

eg: if(a[i] == a[i+1])

---

8) Code to accept N numbers from keyboard and print all elements after reversing them.

To reverse all elements, swap elements with opposite side 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] }

---

9) Let array a[ ] contained N elements, now find whether array is symmetric or not?

Symmetric means: if values remain same even after reversing the array

logic: compare first & last elements, second & previous of last, … until i<j as above shown.

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

---

10) Code to accept N values from KB and print each number's multiples as shown below (use nested-loop)

ip: 14 16 13 11

op: 14 => 1, 2, 7, 14

16 => 1, 2, 4, 8, 16

13 => 1, 13

---

11) Code to accept N values to array and print reverse of each number (use nested loop)

ip: 123 21 529 1312 65781

op: 321 12 925 2131 18756

---

12) Code to accept N values to array and print palindrome

ip: 123 121 529 1312 656

op: 121 656

---

13) Code to accept N values to array and print only primes

ip: 11 17 21 31 15

op: the primes are: 11, 17, 31

---

14) 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 odd and even values separately in the array. (first print all odds and then evens)

```
void main()
{   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[1] );
    -------
    -------
}
```

---

15) Let array a[ ] initially contained list of 10 factorials, for example a[0] contained 1 as 0!, a[3] contained 6 as 3!, a[4] contained 24 as 4!, etc. Now our job is to scan N from keyboard and print its factorial using array.

ip: 4 ip: 5

op: 24 op: 120

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]…
1	1	2	6	24	120	720

```
void main()
{   int a[10]={ 1, 1,  2, 6, 24, 120, 720, 5040 …};   // pre-initialization of array
    int N;      // the input value must be below 10, if not then show an error.
    ----        // here substitute 'N' value in the array to get factorial
}
```

---

16) 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
```

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");
```

---

17) code to scan 5 odd values, while scanning, if input user entered 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.
while( count<5)
{       scan(N);
        if( N%2==1)
        {       a[ count]=N;    // inserting N value into array
                count++;        //  as one odd value inserted into array,  so incrementing 'count'.
        }
}
```
here write code to check and print all primes in the array.

---

18) At theatre, a movie rating data collecting from several audience, the input rating is one by one until last value is -1, the values ranges from 0 to 5. Now find highest count of rating value in 0 to 5. (not average)

Logic:

1) Take array A[] with size 6, here A[0] for counting 0 ratings, A[1] for counting 1 ratings, etc.
2) initialize all cells of array with zero. This is like "int A[6]={0}"
3) scan rating(R) one by one from K.B
4) if(R==-1) stop scanning .
5) if rating R is 3 then increment A[3] cell, if rating R is 4 then increment A[4] cell
for this increment array cells like A[R]++ (this is as shown in below picture)
6) after scanning loop, print biggest of array, which is highest count of rating.

input rating is : 2 4 3 3 3 2 4 3 3 -1 (-1 is end of input)
output : Highest rated number is: 3 (3 given 5 times, it is highest count)

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]
0	0	2	5	2	0

```
void main()
{       int  a[6]={0}, r, i ;
        while(1)
        {       printf("enter rating  ( at end type -1 ) :");
                scanf("%d", &r);
                if(r==-1) break;
                a[r]++;
                ----
```

---

19) In a school, the teacher is announcing marks of each student, the maximum marks is 10. Now our job is to find how many students got 0-marks, 1-marks, 2-marks, 3-marks, etc. The input is reading marks one by one until last input value is -1, later print output as shown below

input: 4↵ 5↵ 5↵ 0↵ 4↵ 4↵ 9↵ 4↵ 0↵ 4↵ 4↵ -1↵

output: 0 marks obtained by 2 students
4 marks obtained by 6 students
5 marks obtained by 2 students ….

logic: take array **"int a[11]"** and initialize all cells with zero, while scanning input marks, increment array cells as **"a[m]++"** where 'm' is input marks of students. Finally, print all non-zero cells as output.

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]…
2	0	0	0	6	2	…

eg: a[0] used to count how many people got zero marks
eg: a[1] used to count how many people got one marks
eg: a[2] used to count how many people got two marks

---

20) Code to scan 5 values one by one and finally print all such 5 values in ascending, the input values ranges in between 0 to 100. (if input wrong by mistake N< 0 or N>100 then don't take into array)

input: 12 8 5 2 15
output: 2 5 8 12 15

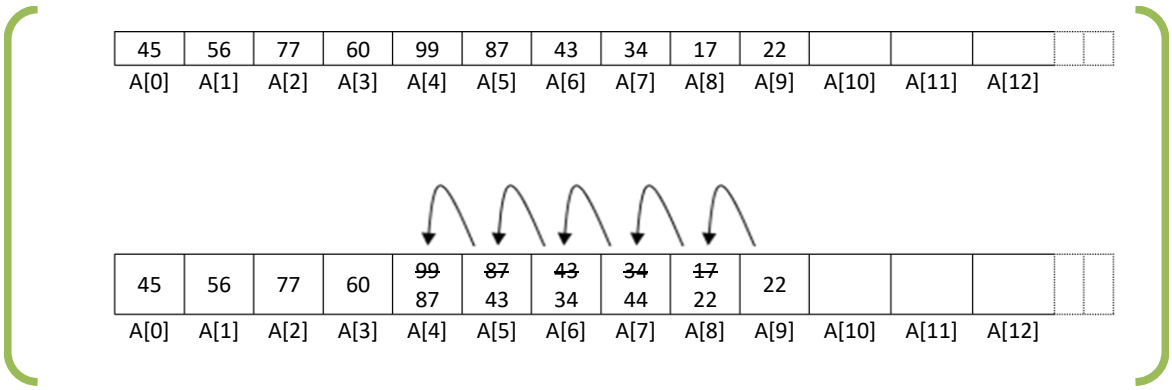
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]	a[12]	a[13]	a[14]	a[15]	a[16]…
0	0	1	0	0	1	0	0	1	0	0	0	1	0	0	1	0

1. first take array A[] with size 101 and initially all cells are zero. (0 indicate absent, 1 indicate present)
2. scan N from keyboard
   now fill array cell with 1, where cell index is N. A[N]=1. (1 indicates present)
3. now print array index (i) where A[i]==1, automatically this output will be in ascending order.

21) Deleting k^{th} element in the array

✦ to delete element in the array, shift all elements one position back from deleting element's position.
✦ the following array contained 10 values, and it gives demo how to delete 5^{th} position element.
✦ The following code deletes 5^{th} 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.
```



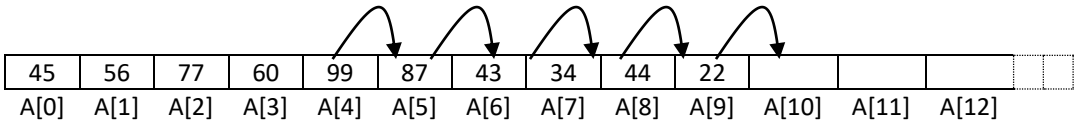
Now write a program, where accept N values from keyboard and delete k^{th} element in the array (k<N) later print all elements after deleting k^{th} element.

---

22) Inserting a new element at k^{th} position

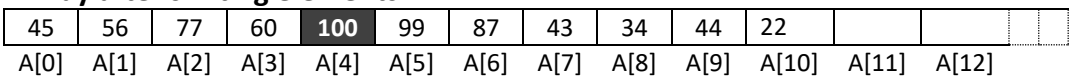
Code to insert a new element at k^{th} position in an existing array of N elements where k<N

To insert a new element at A[k], shift all existing elements of A[k], A[k+1], A[k+2]...A[N-1] to the right side by one position, so that we get a gap at A[k], where new element can be inserted.



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:
```
for(i=n; i>=k i--)    // shifting elements to right side
{     a[i]=a[i-1];
}
a[i]=new;             // inserting new element  at a[k]
n++;                  // as one new element inserted, so increase count.
```

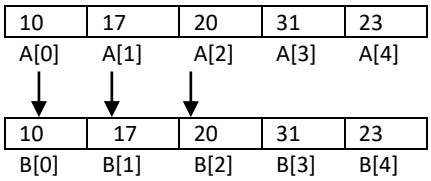
---

23) 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.

ip: a[ ] = {10, 17, 20, 31, 23 };
b[ ] = {10, 17, 20, 31, 23 };
op: yes, both are equal

ip: a[ ] = {10, 17, 20, 31, 23};
b[ ] = {10, 17, 20, 23, 31};
op: no, both are not equal

the comparison as given below figure



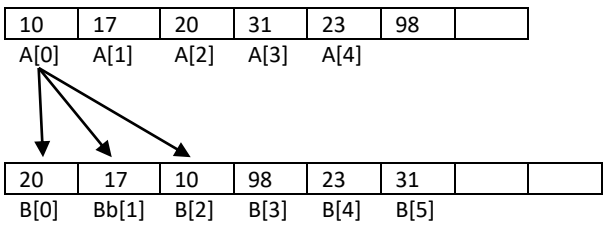
---

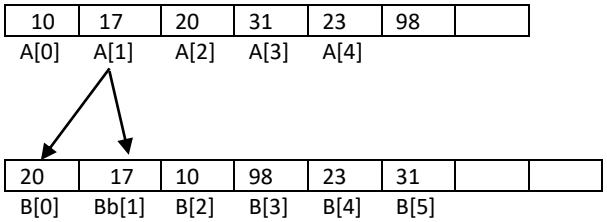
24) Let two arrays have N elements each, now find both arrays have same values or not (values may be in any order)
(Let us take both arrays have same count of N values each without duplicate in the same array)

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





---

25) 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 B)
3) print A[] U B[] (A union B) → A+B-A

---

26) Let array a[ ] contained N values, count frequency of each number (how many times repeated)

ip: 112 221 200 121 879 112 221 112 221 221

op: 112 repeated 3 times
221 repeated 4 times

---

27) 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.

---

28) 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.

---

29) 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 ($(A_0-M)^2 + (A_1-M)^2 + (A_2-M)^2 + (A_3-M)^2 + \ldots + (A_n-M)^2 / N$)

Home Work

---

30) Code to fill array with for first 20 Fibonacci series values and later print them. Let initialize array with first two numbers of Fibonacci and remaining numbers generate by adding like a[2]=a[0]+a[1], a[3]=a[1]+a[2],… Finally print all such 20 Fibonacci numbers.

```
void main()
{   int a[20]={0,1};    // first two values of series are initialized
    ----------
}
```

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	…		
0	1	1	2	3	5	8	13	21	…		

==========

31) Code to find Nc_R (where N or R <10). The list of factorials already calculated and stored in the array, so substitute input value(N,R) in the array to get desired output value.

ip: 4 ip: 9

op: 24 op: 362880

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	A[6]	a[7]	a[8]	a[9]
1	1	2	6	24	120	720	5040	40320	362880
0!	1!	2!	3!	4!	5!	6!	7!	8!	9!

```
void main()
{  long int a[]={1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880};   // initializing with factorials
   scan input N,R here
   substitute N,R values in the array-cell to get desired value and print result.
}
```

==========

32) A shop keeper sells 10 items from his stock inventory, item's code ranges from 1 to 10, and customer buy items by asking item code and quantity, this is as given below, finally calculate bill amount

Input: Enter item code & quantity: 2 5
Enter item code & quantity: 4 2
Enter item code & quantity: 1 1
Enter item code & quantity: 0 (stop when item code zero)

output: total bill calculated as 5*10.30 + 2*9.20 + 1*44.50

Note: Assume the sample price list of each item already given as below, use this array to find bill

Price[0]	Price[1]	Price[2]	Price[3]	Price[4]	Price[5]	Price[6]	Price[7]		Price[10]
(0)Dummy	44.50	10.30	14.50	9.20	14.50	34.80	10.00	….	90.00
No item	Item1	Item2	Item3	Item4	Item5	Item6	Item7	….	Item10

---

33) Let there are N input values (N<100) and all values lie in between 1 to 20, now count frequency of each number (how many times each value is repeated, we already did this program before, but this logic is different)

ip: 5, 5, 6, 7, 5, 14, 7, 9, 7, 7, 8, 8, 8, 11, 10, 17, 17, 17, 17, 17, 20.

op: 5 → 3 times
6 → 1 times
7 → 4 times ….

step1: Let, N values scanned to array a[]

step2: Take another array to store frequency of each number, it is like "int fr[21]={0}".

step3: Now increment cells of fr[ ] where its index is a[i]. This is like fr[a[i]]++; for i=0 to N

step4: now print all cells of fr[] which values are>0. This is as given below array

Dummy	0	0	0	0	3	1	4	3	-	-	-	-	-	
fr[0]	fr[1]	fr[2]	fr[3]	fr[4]	fr[5]	fr[6]	fr[7]	fr[8]	fr[9]	fr[10]	fr[11]	fr[12]	fr[13]	fr[14]---

---

34) A movie rating data is collected from audience at theater, the rating values ranges from 0 to 9, collected from several people one by one until last input is -1, now our job is to sort all ratings and print as given below.

ip: 2 4 6 2 5 8 9 3 3 2 1 0 9 0 0 1 1 -1

op: 0 0 0 1 1 1 2 2 2 3 3 4 5 6 8 9 9

Note: do not use any sorting technique, take array with size 10 and initialize with zero in every cell of a[], now increment a cell where its index is rating, later print all of them.

---

35) Code to accept values one by one until last input value is zero, if input value is prime then take into array (insert into array) , if not prime then skip such number and finally print all values in the array(primes)

ip: 11 12 30 20 19 17 14 45 11 13

op: 11 19 17 11 13

---

36) Code to accept N values into array, later check whether one number divides with any other number in the array or not? Finally count such divisible numbers (use nested loop)

ip: 5 15 4 28 11 // 15 divides with 5, 28 divides with 4

op: count = 2 (the numbers are 15, 28)

---

37) Code to fill array with prime numbers from 2 to 1000.

The output filled array 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(i=3; i<100; i++)
    {     -----
    }
    -----
}
```

this is fastest technique to check prime-ness of one number with previous primes.

---

38) Code to generate 10 random numbers, these must be distinct and must lie in between 100-200.

Logic: the library function **rand()** generates the rand numbers in between 1-65535.

for example: k=rand()%100+100; → generates a number in between 100-200.

note: rand() function may generate duplicate values, avoid duplicates while collecting numbers into array.

hint: use header file 'stdlib.h' for rand() function

---

39) Let array contained N values, in which some elements may repeated more than once, any way print only unique elements(not repeated elements). Let all values in the array are +ve (a[i]>0)

ip: 14 16 18 20 14 38 21 16 14

op: 18 20 38 21

```
for(i=0; i<N; i++)              // this loop is to check all elements in the array
{   if(a[i]==0) continue;       // already counted in previous iterations of loop, so skip it.
    for(j=i+1; j<N; j++)        // this loop is to check duplicate , compares a[i] with all other in the array
        if(a[i]==a[j])
        {   count++;
            a[j]=0;             //  delete a[j] from array, to avoid next count
        }
        -------
}
```

---

40) Code to accept 5 values one by one from keyboard, while scanning values, the next input should not be less than previous value, if user entered by mistake then reject it. Observe following ip/op

input: enter value 1: 12
enter value 2: 16
enter value 3: 7
the input value '7' is 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

---

41) Code to accept 5 values one by one from keyboard, do not allow duplicate 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

---

42) 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

---

43) Following program simulate the bus reservation process, here repeatedly tickets are issued one by one to passengers, at end process, the seat number -1 is entered as end of session. Finally print list of reserved seats.

step1: take array: int a[40]; // let 40 is the maximum number of seats
step2: Initialize all locations of array 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, go to step5;
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

---

imp 44) 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

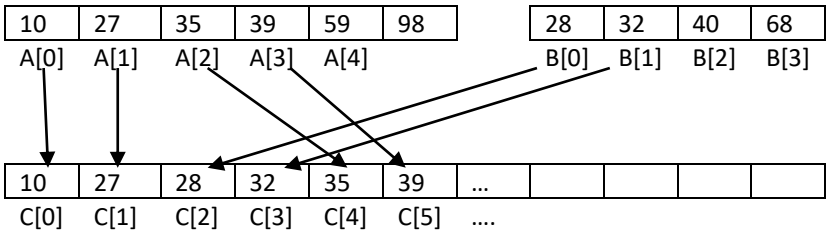
---

imp 45) 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 };

B[ ] = {28, 32, 40, 68 };

op: C[ ] = {10, 27, 28, 32, 35, 39, 40, 59, 68, 98 }



step1: let array A[] contained n1 elements already in sorted order (input in sorted order)

step2: let array B[] contained n2 elements already in sorted order (input is 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 copying them to C[], as given below

```
while(i<n1)          // after above loop, still if i < n1 means some elements left in A[], so copying to C[]
    C[k++]=A[i++];
while(j<n2)          // after above loop, still if j < n2 means some elements left in B[], so copying to C[]
    C[k++]=B[j++];
```

---

46) Write a program to increment given date by 'n' days

ip: enter date: 31 12 2019 ↵

enter no.of days to increment: 365↵

op: 31 12 2020

step 1: fill (initialize) all month's days in an array like a[13]={0, 31, 28, 31, 30, 31,....}

step 2: let date is scanned to (d, m, y)

step 3: a[2]=28+(y%4==0); // if leap year, then 28+1

step 5: take loop, repeat n times as shown below;

```
for(i=0; i<n; i++)              // loop increments date by one day in every iteration;
{   d++;                        // incrementing day by one-day
    if( d==a[m] )               // if 'day' reached to end of month, then shift to next month
    {   d=1; m++;               //  shifting to next month
        if(m==13)               // if month is reached to end of year then shift to next year
        {   m=1;  y++;                  // shifting to next year
            a[2]=28+(y%4==0);       // if leap year 28+1; otherwise 28+0
        }
    }
}
printf("Date after incrementing is %d %d %d",  d, m, y);
```

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 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;
        ----
        k=big(a,b);
        k=big(k,c);          //  we can also call like: k=big( big(a,b), c )  → embedded calling
        -----
}
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 for %f salary is  %f", salary, tax);
}
float findTax(float  salary)
{       -------
        -------
}
```

---

4) Write a fn to calculate x^y value, where 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 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 main() function and check this 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)
{       -------
        -------
}
```

---

6) Write a fn called **sumOfCubes(),** which returns the sum of cubes of all digits of a given argument **N**, later write main() fn and check given number is Armstrong or not? fn proto-type is: **int sumOfCubes(int);**

ip: 135 ip: 153 ($1^3+5^3+3^3 \rightarrow 1+125+27 \rightarrow 153$)

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;
}
```

---

7) Write a fn called **reverse(),** which returns the reverse of a given argument **N**, later write a main() fn and check the given **N** is palindrome or not? fn proto-type is: **int reverse(int);**

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;
}
```

---

8) Write a fn called **sumOfDivisors(),** which returns the sum of all divisors of given 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 given 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

---

9) 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 )
{       ------
        ------
}
```

---

10) 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);**

---

11) Write a fn called **sumOfCubes(),** which returns the sum of cubes of all digits of a given argument **N.**

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

---

12) Let at main() function there is an array A[] with 10 predefined values, now yourself write **isPrime() f**unction body and its call statement to count primes in the array. The function takes N 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 )
{       ---------
}
```

---

13) Let at main() function there is an array A[] with 10 predefined values, now yourself write **reverse() f**unction body and its call statement to count palindromes in the array. The function takes 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;
        -------
        -------
```

---

14) Write a fn called **printTable()** which prints multiplication table upto 10 terms. (this fn returns nothing) this function takes 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)
{       --------
}
```

---

15) using above **printTable() fn**, write a main() fn and print all multiplication tables from 1 to 20.

---

16) 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)
{       ----------
}
```

---

17) Write a fn called **printAllDivisibles(),** which prints all divisibles of given N.
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;
                -----
        }
}
-------
```

---

18) Write a fn called **sumOfSquares(),** which returns sum of squares of 1 to 10. ($1^2 + 2^2 + 3^2 + 4^2 \ldots + 10^2$)

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)
{   -----------
}
```

---

19) 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/failed. If he got >=50 in 2 subjects then print "pass" or else "fail".

```
void main()
{       int m1,m2;
        m1=readMarks();
        m2=readMarks();
        ------
}
int readMarks()
{       -----           // here scan marks of one subject and return if input is valid, if not then again scan.
}
```

---

20) 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)

---

21) 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.

---

22) 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
}
```

---

23) Write a fn "**nextPrime()",** which returns the next prime of given argument N. If N 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);**

Home Work

30) Write a function which takes 3 arguments and returns the big. Later check by calling from main() fn.

---

31) Write a function which takes 2 arguments as month & year, and returns no.of days in that month. Later check by calling from main() fn with sample input values.

---

32) 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 it.

---

33) Write a fn to check given date is valid or not?, the function takes date(d,m,y) as arguments and returns the true/false value. Later write a main() fn where scan date from keyboard and print how many days left in that month(from given day). If user entered invalid date then scan again and again until a valid date entered.
ip: 5 1 2000
op: 31-5→26 days left

```
void main()
{       int d,m,y;
        -----
}
int isValidDate(int d, int m, int y)
{       ----------
}
int getDaysInMonth(int m, int y)
{       ----------
}
```

---

34) Write a fn called **getAreaOrPerimeter()** which takes radius and flag as arguments and returns area/perimeter of circle. (If flag is 'A' then it returns Area of circle, if flag 'P' returns Perimeter)

Function proto-type is: float getAreaOrPerimeter(float radius, char flag);

the main() fn already given with radius = 5

```
void main()
{       float area, radius=5, perimeter;
        area=getAreaOrPerimeter(radius , 'A' );          // function call-1
        printf("\n area  is : %f", area);
        perimeter=getAreaOrPerimeter(radius , 'P' );     // function call-2
        printf("\n perimeter is : %f", area);
}
float  getArea( float radius, char flag )
{       -------
}
```

---

35) Write a function which takes 3 arguments as time and return in seconds format, Later write main() fn where scan 2times from KB and print they are equal or not? Take variables as: (h1,m1,s1), (h2,m2,s2)

---

36) Write a function which takes time in seconds format, and return Hours, minutes and seconds value as given below.

```
void main()
{   int N=7290 , h, m, s;
    h = getTime(N, 'H' );
    m = getTime(N, 'M' );
    s =  getTime(N, 'S' );
    printf("time  is  %d : %d : %d" , h, m, s);
}
```

---

37) 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)
{   ----------
}
```

---

38) 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

---

39) 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 )
{       --------
        --------
}
```

Recursion

(don't use any loop control structure in the following examples)

1) Fill the body of recursive function **show()** to print 1 to 10 numbers.

```
void main()
{       show(1);
}
void show( int i )
{       -----
        -----
}
```

---

2) 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 i )
{       -----
        -----
}
```

---

3) Fill the body of recursive function **show()** to print 1 to 10 and 10 to 1 numbers.

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

```
void main()
{       show(1);
}
void show( int  i)
{       -----
}
```

---

4) Fill the body of recursive function **show()** 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

```
void main()
{       int N=100;        // or else scan 'N' value from keyboard using scanf() statement
        show(N);
}
void  show(int N)
{       ------
}
```

---

5) 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)
{       ------
        -----
}
```

---

6) Write a recursive function to print multiplication table up to given number of terms

ip: 8

op: 8*1=8
8*2=16
8*3=24
……
8*10=80

```
void main()
{       int N=8;
        show( N, 10 );          // 'N' is table number, and 10 is number of terms to print
}
void show( int N, int i )       // 'N' represents which table to print,  'i' is like looping
{       ---------
        ---------
        printf("\n %d * %d = %d", N, i, N*i);
        --------
}
```

---

7) Write a recursive function to print following output 1, 2, 4, 8, 16, 32, 64 up to N terms.

```
void main()
{       int N=10;
        show(1, N);       // 'N' is number of terms to print, where '1' is starting value of series.
}
void show(int P, int N)  // 'N' is to count down from 'N' to 0, where 'P' raises its value to 1, 2, 4, 8, ….
{       -------
        -------
}
```

---

8) Write a recursive function to print Fibonacci series up to N terms

ip: N=11 (11 terms)

op: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55

```
void main()
{       int N=11;
        show(0,  1,  N);        // 0,1 are first two terms in the fibo series, N is number of terms to print
}
void show(int x, int y, int N)
{       ---------
        print(x);
        show(y,  x+y, N-1);     // for next recursive call of fibo() function.
}
```

---

9) 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( -- );
        ------
}
void printRow(int N)
{       -----           // prints each row here
        -----
}
```

---

10) 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)
{       ----                            // here stop condition
        return 2+find(N-1);             // next recursive call.
}
```

---

11) 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 )
{       ----
        ----
}
```

---

12) 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, proved");
        else  printf("no, not proved");
}
int find( int N )
{       ----                    // stop when N becomes 0 then stop.
        ----
}
```

---

13) 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)
{       ------
        ------
}
```

---

14) 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)
{       ----
        ----
}
```

---

15) 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)
{       --------
        --------
}
```

---

17) 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.
{       ------
        ------
        ------
}
```

16) find best code of 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);
}
```

Home Work

---

27) Fill the body of recursive function **show()** to print 1 to 10 and 9 to 1 numbers.

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

```
void main()
{       show(1);
}
void show( int  i)
{       -----
        -----
}
```

---

28) write a recursive function to print binary number of given **N**. Let **N** is 13 then the binary of 13 is 1101

Logic: print 13%2 → 1
print 6%2 → 0
print 3%2 → 1
print 1%2 → 1

Here print remainders of N%2 from bottom to top, remember N value is changing to N=N/2 in every call.

---

29) Write a function to print odd numbers 1, 3, 5, 7, 9, 11, 13, … < N

```
void main()
{       int N=16;
        show(N);
}
void show(int N)
{       -----
        -----           // check N, whether it is odd or even, if odd then print(N)
}
```

---

30) Write a function to print 1, -2, 3, -4, 5, -6, 7, …N times

```
void main()
{       int N=13;
        show(N);
}
void show(int N)
{       -----
        -----           // check N, whether it is odd or even, if odd then print(–N) or else print(N)
}
```

---

31) 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)
{   ----------
    ----------
}
```

---

32) 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

---

33) 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)
{   ----
}
```

---

34) 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
}
```

---

37) Write a function to find sum of all digits in a given number, the function takes N as argument and returns sum of all digits.

ip: 2345 ip: 456 ip: 23456

op: 14 (2+3+4+5) op: 15 (4+5+6) ip: 20 (2+3+4+5+6)

hint: return N%10+find(N/10)

---

38) Write a function to find sum of odd digits only.

ip: 2345

op: odd digits sum = 8 (3+5)

---

39) Write a function to print 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.

---

40) write a recursive function to print first odd digit of a given number, if odd not found then return -1.

ip: 2345 ip: 456 ip: 486

op: 3 op: 5 op: -1

---

35) Write a program to find GCD of two numbers. (GCD/HCF → Greatest Common Divisor)

ip: 12, 18

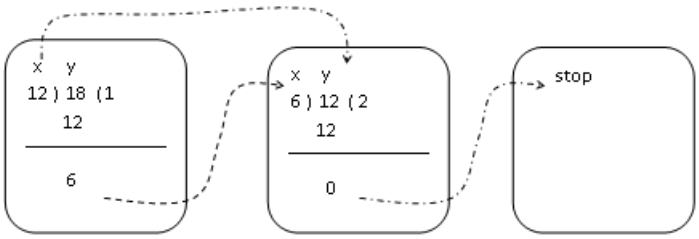
op: 6

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.



---

36) 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)

---

41) Write a program to find LCM of 3 numbers

ip: 12 18 45

op: 180

1. Let x, y, z are three input numbers
2. Start finding **LCM** of three with first factor 2
3. Now divide each x, y, z with 2, and decrement all divisible numbers to quotient obtained in the division.
   For example, if x is divided with 2 then decrement x=x/2
4. If any x, y, z is divided in step3 then take 2 as factor in **LCM**. (LCM=LCM*2)
5. Now repeat step-3, step-4 as long as 2 divide any x, y, z
6. if 2 no longer divided, then next try with next factors 3, 4, 5…etc,
   repeat this process until any of these (x, y, 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

---

42) Write a recursive function to print value of array, let array a[] contained N values.

```
void main()
{       int a[]={10, 43, 3, 11, 6, 5, 22, 60},N=8;        // array contained 8 values.
        show(a ,N);
}
void show( int *p, int N)
{       -----
        printf("%d ", *p);                  // don't change this line
        -------
}
```

---

43) Write a recursive function to print following output

```
op: APPLE
    PPLE
    PLE
    LE
    E
void main()
{       show("APPLE");
}
void show(char *p)
{       ------
        printf("\n%s", p);
        show(….);
}
```

---

44) 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
{       -------
}
```

---

45) 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…		

---

46) 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…					

---

47) 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 any number of values indirectly through address, this concept is called returning more values through pointers or 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 takes radius of circle as argument and returns area & perimeter

```
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) Correct the following **incrementBy10()** function and its call statement.

```
void main()
{       int n=25;
        incrementBy10(n);
        printf("n value is  %d ", n);          // expected output is 35 not 25.
}
void incrementBy10(  int  p )
{       p=p+10;
}
```

---

6) fill the following function body, it should replace a=a+b and b=|a-b| for any a>0 , b>0

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( --- , --- )
{   ----
    ----
}
```

---

7) 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;
}
```

---

8) 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( ----, ----)
{
        -----
}
```

---

9) Complete the body of following **swap2()** function for swapping a , b.
the **swap1()** taking help of **swap2()** for swapping values. (There is litter 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( ----, ----)
{
    -----
}
```

---

10) 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)
{
        --------
}
```

---

11) Complete the code of **scanTwo() and printTwo()** functions which has to scan and print two values .

```
void main()
{   int a,b;
    scanTwo( ---- );
    printTwo( ---- );
}
void scanTwo( ---- )
{   -----          // here scan two from keyboard and store into a,b of main() fn.
}
void printTwo( --- )
{       -----      // here print  a , b values of main() fn
}
```

---

12) 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
}
```

---

13) 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( N , &big , &small );
        printf("\n big is  %d , small is  %d ", big , small );
}
void findBigSmall( int  N,  int  *pBig,  int  *pSmall)
{       -------
}
```

Logic to find big digit in N.

```
while(N>0)
{       rem=n%10;
        if(big<rem) big=rem;
        if(small>rem) small=rem;
        n=n/10;
}
```

---

14) 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;
```

---

15) 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
07 50 40
op: 18 31 30

```
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;
```

---

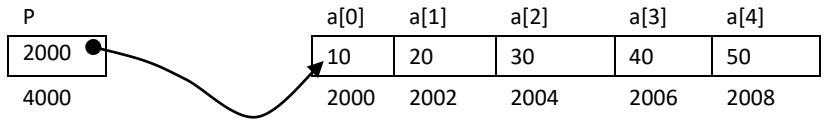
16) Write a function to increment date by one day, this function takes day, month, and year as arguments and return the incremented date.

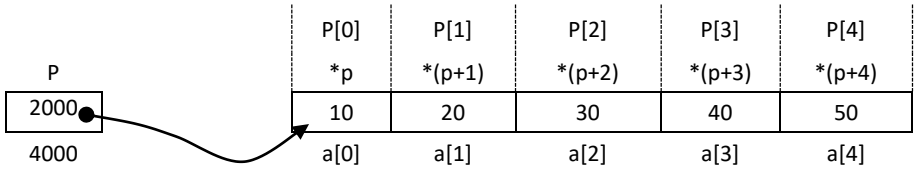
```
void main()
{       int d,m,y;
        scanf("%d%d%d", &d, &m, &y);
        increment( ---- );
        printf("\n output: after incrementing the date is %d  %d  %d", d, m, y );
}
void increment( ------- )
{       --------
        --------
        --------
}

// 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++;
        if( d>arr[m] )
        {       d=1;  m++;
                if(m==13)
                {       m=1; y++;
                }
        }
```

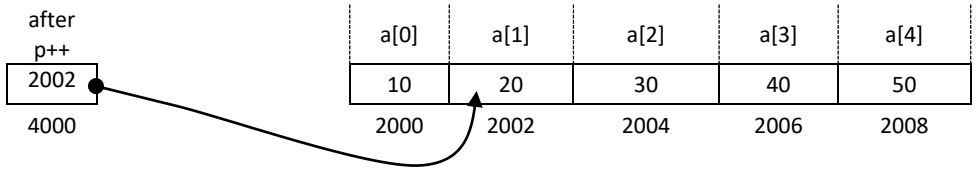
Pointer to Array

following program explains how to access array elements through pointer

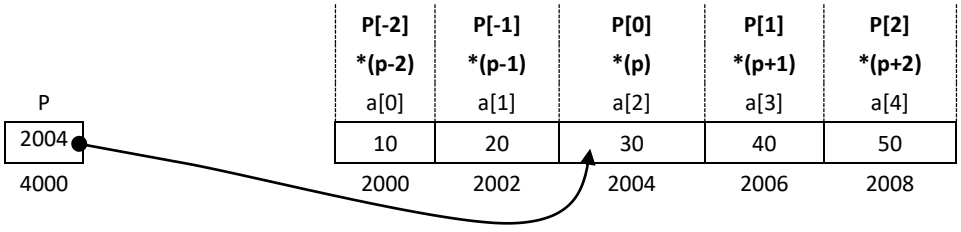
```
void main()
   {      int arr[5]={10,20,30,40,50};
          int *p;
          p=&a[0];   or   p=a;
          for(i=0; i<5; i++)          //method-1
                printf("%d  ",  *(p+i)  or p[i]  );
          or
          for(i=0; i<5; i++)          //method-2
          {      printf("%d  ",   *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) ) ;
}
```

17) Complete the code of function which has to increment each value of array 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 )
{       -------
}
```

---

18) 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

---

19) 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

---

20) Write a function to search given value in the array exist 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] , key );
    if(bool==1) printf("element exist");
    else printf("element not exist");
}
int search( ----- )          // this function returns bool value
{   ------
}
```

---

21) 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 elements 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  dest[], int source[], int n)
{   --------
    --------
}
```

---

22) Write a function called find (), which takes array base address & number of values in the array and 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.
        ----
```

---

23) 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 A[]  )
{   int i;
    for(i=1; i<=A[0];  i++)
        printf("%d   ",  A[i] );
}
```

---

24) Write functions called **readMe()** & **writeMe()**, the function **readMe()** scans 5 values from keyboard and stores into array and the function **writeMe()** prints such 5 values on the screen.

ip: 12 32 43 57 11
op: 12 32 43 57 11

```
void main()
{   int a[5];
    readMe( --- );
    writeMe( --- );
}
void readMe( --- )
{   ------
    ------
}
void writeMe( --- )
{   ------
    ------
}
```

---

25) Write functions called **readMe()** & **writeMe()**, the function **readMe()** scans N values from keyboard and assigns into array and the function **writeMe()** prints such N values on the screen.

```
void main()
{   int a[20], n;              // Let us say n<20
    readMe(&a[0] , &n );
    writeMe(&a[0] , n );
}
void readMe( int p[ ] , int *x )
{   ------
    ------
}
void writeMe( int p[ ] , int n )
{   ------
    ------
}
```

---

26) Extension to above programs, read two array of 5 values to each and check two array inputted same values in same order or not? Let two arrays are A[] and B[]

ip: 12 32 43 57 11 for A[]
12 32 43 57 11 for B[]
op: yes

ip: 12 32 43 57 11 for A[]
22 42 14 57 11 for B[]
op: no

<u>the main() fn is as follows</u>

```
void main()
{   int a[5], b[5], bool;
    readMe( --- );          // read 5 values to A[]
    readMe( --- );          // read 5 values to B[]
    bool=compare(---,---);
    if(bool==1)  printf("yes");
```

```
    else  printf("no");
}
void readMe( --- )
{   ------
    ------
}
int  compare( ---- , ---- )
{   ------
    ------
}
```

---

27) 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 B)
3) print A[] U B[] (A union B) → A+B-A

note: 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 program.

---

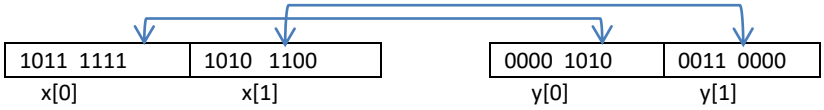
28) Write a function called findBigSmall(), which takes array base address & number of values in the array and to return the big & small of them.

Later write a main() function to check it (also write a function for scanning input values to array).

```
void findBigSmall( int a[] , int n , int  *pBig , int  *pSmall );
void scanArray( int a[], int *pn);        // this scans 'n' values to array
```

---

29) 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);



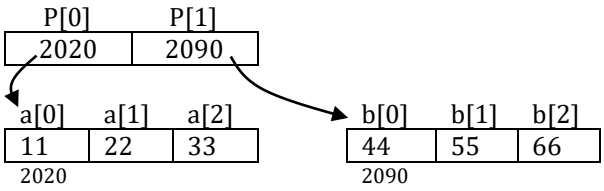

```
// this swap() fn, swaps any type of data (it can be int/float/char/long-int)
   void swap(void *p, void *q, int size);
   {  char *x, *y;
      x=(char*)p;          // type casting, converting  void*  to  char*
      y=(char*)q;
      for(i=0; i<size; i++)
      {  --------
         --------          // 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 );
}
```

---

30) 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.
          printf("\n");
    }
}
```



---

31) Complete the following code to print 2 array of values, the output should be

```
    11   22   33
    44   55   66
void main()
{   int *p[2];
    fill( ----- );
    for(i=0; i<2; i++)
    {     for(j=0; j<3; j++)
              printf( -----);
          printf("\n");
    }
}

void   fill( ---- )
{   int *x, *y;
    x=(int*)malloc(sizeof(int)*3);
    y=(int*)malloc(sizeof(int)*3);
    x[0]=11, x[1]=22, x[2]=33;
    y[0]=44, y[1]=55, y[2]=66;
    ------
}
```

Characters & Strings

01) Code to accept a character and find whether it is lower case alphabet/upper case alphabet/digit /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 alphabet 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.

```
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 alternative characters.

ip: computer

op: cmue

---

13) Code to accept a string and print following way

ip: computer

op: computer
omputer
mputer
puter
uter
ter
er
r

---

14) Code to accept a string and print following way

ip: computer
op: computer
compute
comput
compu
comp
…..

---

15) 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

---

16) 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

---

17) 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

---

18) 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

---

19) 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

---

20) Code to accept a multi word string and convert all first characters of each word to upper-case and remaining to lower case

Ip: all apples are fruits and all fruits are not apples
op: All Apples Are Fruits And All Fruits Are Not Apples

---

21) Code to accept a string and print each character frequency

ip: all are good programmers
op: a-3 times repeated
d-1 time repeated
e-2 times repeated ….

---

22) Check if two Strings are anagrams of each other?

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

---

23) 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

---

24) 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[] )
{   -----
    -----
}
```

---

25) Code to reverse 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 string length

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
'H'	'E'	'L'	'L'	'O'	'\0'				

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
'O'	'L'	'L'	'E'	'H'	'\0'				

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[] )
{   -------
    -------
}
```

---

26) 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 )
{   -------
    -------
}
```

---

27) 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 )
{   -------
    -------
}
```

---

28) Let X , Y are two strings, scan and print whether X>Y or X<Y or X==Y

Note: String 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 can be +ve/-ve. if +ve when X>Y, if -ve when X<Y, Zero when X==Y.

```
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 )
{   -------
    -------
}
```

---

29) Extend above program by ignoring case (the alphabets can be upper/lower case)

Input1: Hello
Input2: heLLo
output: X==Y

---

30) 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);
    printf("after copy, the string is %s", b);
}
int myStrcpy( char *destination , char *source )     // destination=source
{   -------
    -------
}
```

---

31) 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
}
int myStrcat( char *p, char *q )
{   -------
    -------
}
```

---

32) 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								

---

33) 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							

---

34) Write a function called **myStrchr()**, to find and return address of first occurrence of given character, if not exist then return NULL pointer value.

```
void main()
{   char arr[20]=”all books are expensive”;
    char *p;
    p=myStrchr(arr, ‘Z’);
    if(p==NULL)
        printf(“character not found”);
    else
        printf(“character found”);

    p=myStrchr(arr, ‘e’);
    if(p==NULL)
        printf(“character not found”);
    else
        printf(“character found”);
}
char* myStrchr( char *p, char ch )
{   -------
    -------
}
```

---

35) Program to accept multi word string and print each word in reverse order
(write your own version of function to reverse the word, using it, we can solve easily)

ip: all apples are fruits
op: lla selppa era stiurf

---

36) Program to accept a multi word string and find whether specific sub-string is exist or not?

ip: main-string: all apples are fruits
sub-string: are
op: yes

---

37) 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]="C language is function oriented language, C++ is oop oriented language"
    char b[20]="oriented";
    char *p;
    p=myStrstr( a , b );
    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: oriented language, C++ is oop oriented language"

---

38) 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

---

39) 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

---

40) 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

---

41) 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

---

42) 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 = 13
Lines count = 3
Digits count=7

---

43) Write a program to accept an arithmetic simple expression from keyboard and print its sum.
The operators are + - * / % and the values are integers

ip: 10+20 ip: 20*30 ip: 20/10
op: 30 op: 600 op: 2

---

44) 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)

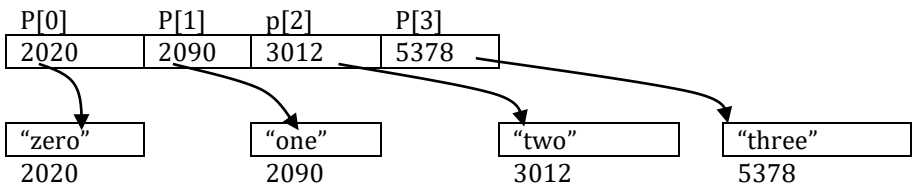
ip: 2983453
op: 29,83,453

---

45) 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.

---

46) 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'					

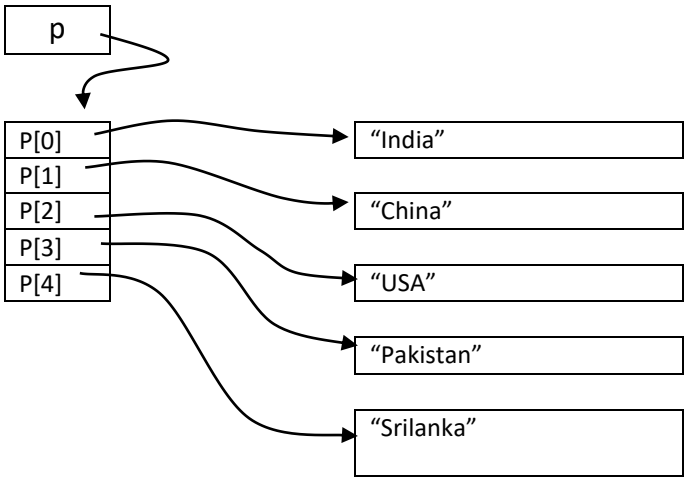
After sorting, the strings are

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'				

---

47) 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]);
}
```

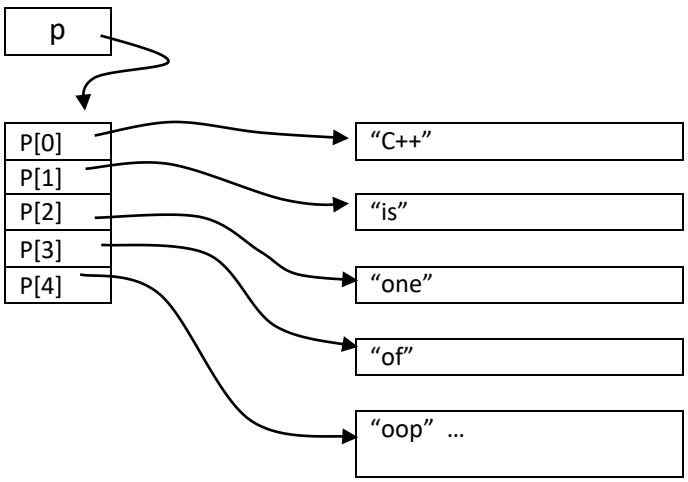


---

48) 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 Cicle
{   int radius;
    float area, perimeter;
};
void main()
{       struct Circle  a;
        scan( &a );          // call by reference
        find( &a );          // call by reference
        show( a );           // call by value
}
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) using read() function.
    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(…..)  { …. }
void  isValidDate( …..)  { …. }
```

---

5) 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);
}
void  readDate(.....)  {  .....  }
void  printDate( .... )    { .... }
void  incrementDate( .....)
{   // sample code for incrementing date for n days (modify this code according 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);
            }
        }
    }
}
```

---

6) 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( ---- )
{    ----
    ----
}
```

---

7) 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;
    int n;
    read( &t );                  // scan time1 (hours, minutes, seconds) at this function.
    n=3600;                      // or else scan N from keyboard
    increment( &t , n );         //  increment  't' by n seconds
    write(t);                    // print time on the screen
}
void read( ---- )
{   ---
    ---
}
void write(---)
{   ----
    ----
}
void increment(---, ---)
{       // sample code for adding two times
        // converting total time into seconds
        n=(h1+h2)*3600 + (m1+m2)*60 + (s1+s2);
        //  distributing  'n'  to  h, m, s format
        h3=n/3600;
        m3=(n%3600)/60;
        s3=(n%3600)%60;
        // modify this code according to  structures
}
```

---

8) 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;
};
-----
-----
```

---

9) This program about two players of the game, where add score, print score, etc.
Complete the code of following functions.

```
struct Game
{       char  name1[30],  name1[30];    // two players names
        int score1,  score2;       // two players scores
};
void accpetNames()
{       struct Game  g;
        setNames(&g  , "Ram",  "Ravi");  // set name of players and also make zero to score1,score2
        addScore(&g, "Ram", 3);           // 3 is points
        addScore(&g, "Ram", 5);
        addScore(&g, "Ravi", 2);
        showScore( g );                   // display score of two players along with names
}
void setNames( … ) { …. }
void addScore( … ) { … }
void showScore( … ) { … }
```

---

10) 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( ---- )
{    ---
}
```

---

11) 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

12) 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

13) 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"},
                            ...
                         } ;
// nonstop 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( ---  , ---)
      {   ------
          ------
      }
```

---

14) 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)	

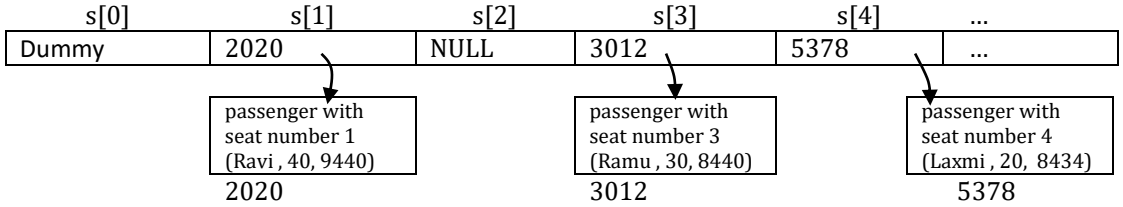
p[0]		p[1]		p[2]		p[3]		P[4]	
15	4	10	3	13	2	7	0	0	
coeff	expo	Coeff	expo	coeff	expo	coeff	expo	coeff	

```
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);
        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[ ])
{       -------
        -------
}
```

---

15) 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];
};
```



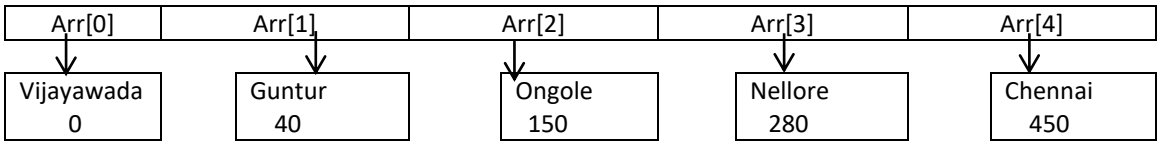
```
void main()
{   struct Passenger  *arr[21]={NULL};       // let total seats are 20 (first is dummy)
    int choice;
    while(1)
    {   printf("\n 1. Reserve ");
        printf("\n 2. Cancel ");
        printf("\n 3. Print ");
        printf("\n 0. Exit ");
        scanf("%d", &choice);                  // 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[]  )
{       -----
        -----
}
```

---

16) 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 write & read 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()
{       writeToFile();
        readingBackFromFileAndPrint();
}
void writeToFile()
{       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 → space
        fclose(fp);
}
void readingBackFromFileAndPrint()
{       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()
{       writeToFile();
        readingBackFromFileAndPrint();
}
void writeToFile()
{       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 readingBackFromFileAndPrint()
{       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 screen: 20   21    22   23   24
        }
        fclose(fp);
}
```

---

03) 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 '^' 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 ^) :");
        while(1)
        {       ch=getchar();               // scanning char by char from keyboard
                if(ch=='^') break;
                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);
}
```

---

04) Let our text file **"input.txt"** contains collections of integers, 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.
hint: to read integers from file, use fscanf(fp, "%d", &n);

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

ip: enter file name : "input.txt"
op: odds count=13

---

05) 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

---

06) 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"**
Text files contain textual data and may be saved in plain text or rich text formats. While most text files are documents created and saved by users, they can also be used by software developers to store program data. Examples of text files include word processing documents, log files, and saved email messages. contact 9440-030405

ip: file name: **"sample.txt"**
op: words count = 55
lines count = 6
digits count = 10

---

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

/* 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);
}

→

After removing comments, the file is

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

void main() { in a,b,c;
printf("enter two values");
scanf("%d%d", &a, &b); c=a+b;
printf("output = %d", c);
}

→

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);
}
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"

void main()
{ int a, b, c;
printf("enter two values:");
scanf("%d%d", &a, &b);
if(a>b && a>c)
printf("big=%d", a);
else if(b>c)
printf("big=%d", b);
else
printf("big=%d", b);
}

Input & output

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);
   }
```