ARRAYS

Q. DAF to input n no. in an array, calculate avg and print those no. greater than average.

ARRAY : collection of homogeneous type of data

datatype array-name [size]
↳ int const.

int a[3];

a[0], a[1], a[2]
↳ index

Memory Allocation:
continuous memory

Initialization:

partial declaration (initialization)

int a[3] = {1, 2};

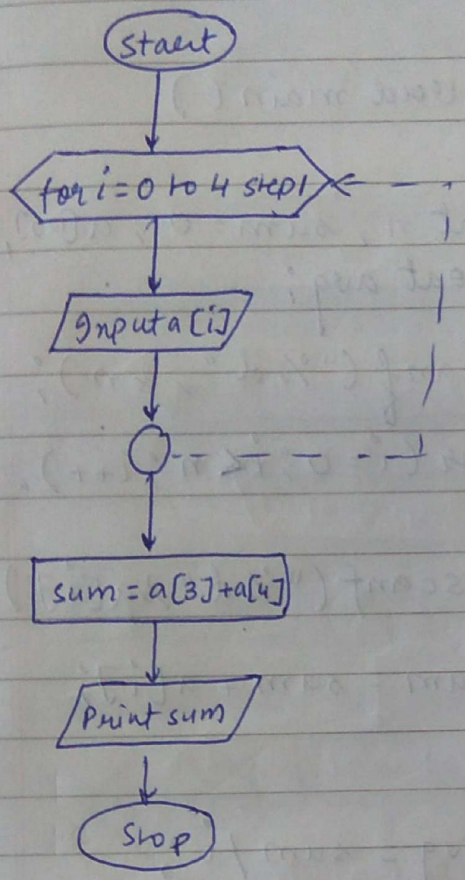
int a[3];
a[0] = 1;
a[2] = 2;

full initialization

int a[3];

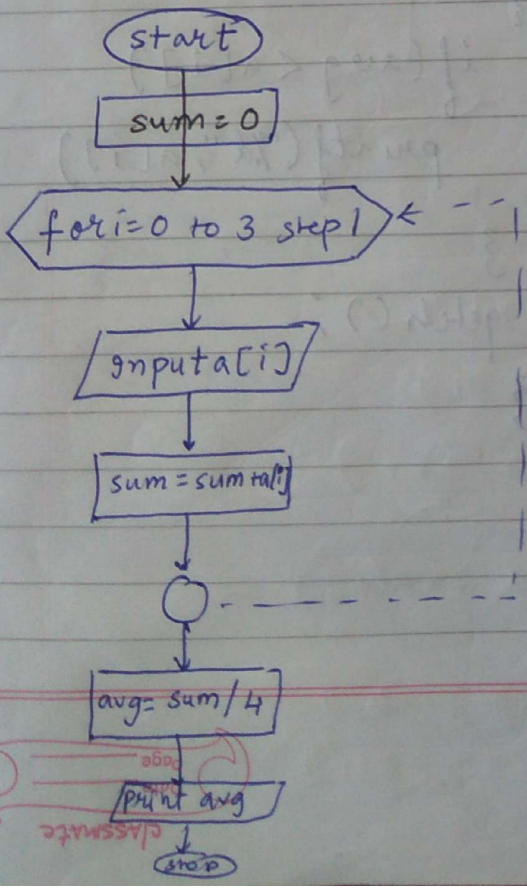
for (i=0; i<3; i++)
{ scanf("%d", &a[i])
}

Q DAF to input 5 no. in an array and print sum of 3 and 4 index variable value.



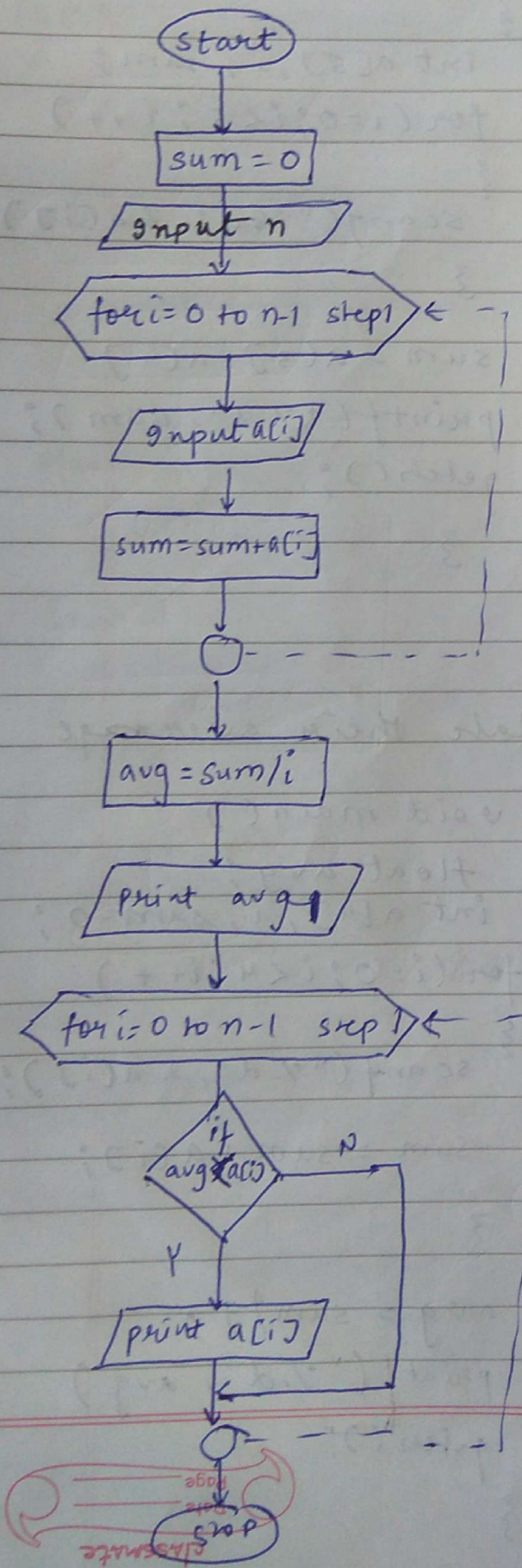

```
void main()
{
 int a[5], i, sum;
 for(i=0; i<5; i++)
 {
  scanf("%d", &a[i]);
 }
 sum = a[3] + a[4]
 printf("%d", sum);
 getch();
 }
```

Q. DAF to input 4 no. an calculate their average



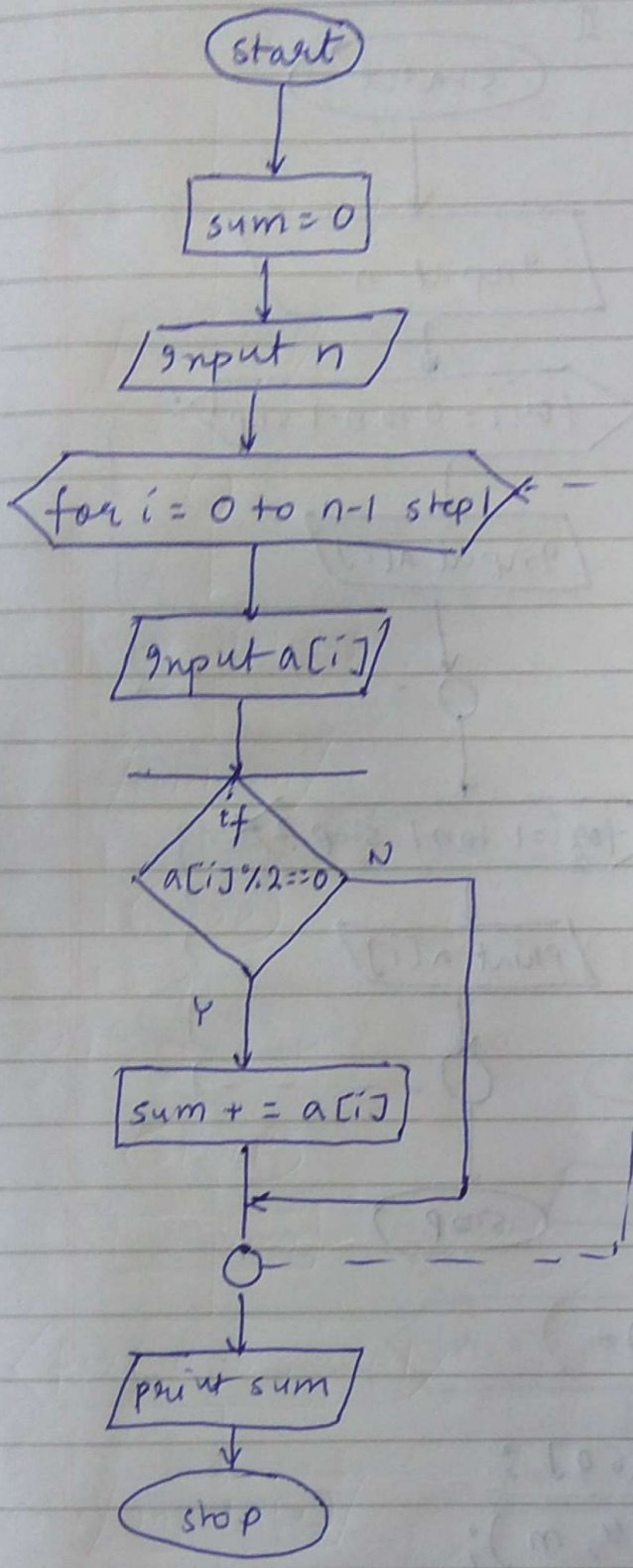

```
void main()
{ float avg;
  int a[4], i, sum=0;
for(i=0; i<4; i++)
{ scanf("%d", &a[i]);
  sum = sum + a[i];
}
avg = sum/4;
printf("%d", avg)
getch();
}
```

Q. Dff to input n no. in an array, calculate their avg. and print those no. which are greater than avg.




```
void main()
{
int n, sum=0, a[100], i;
float avg;
scanf("%d", &n);
for(i=0; i<n; i++).
{
 scanf("%d", &a[i]);
sum = sum + a[i];
}
avg = sum / i;
for(i=0; i<n; i++)
{
  if(avg < a[i])
   printf("%d", a[i])
}
getch();
}
```

Q. DAF to input n numbers and print sum of n even numbers.



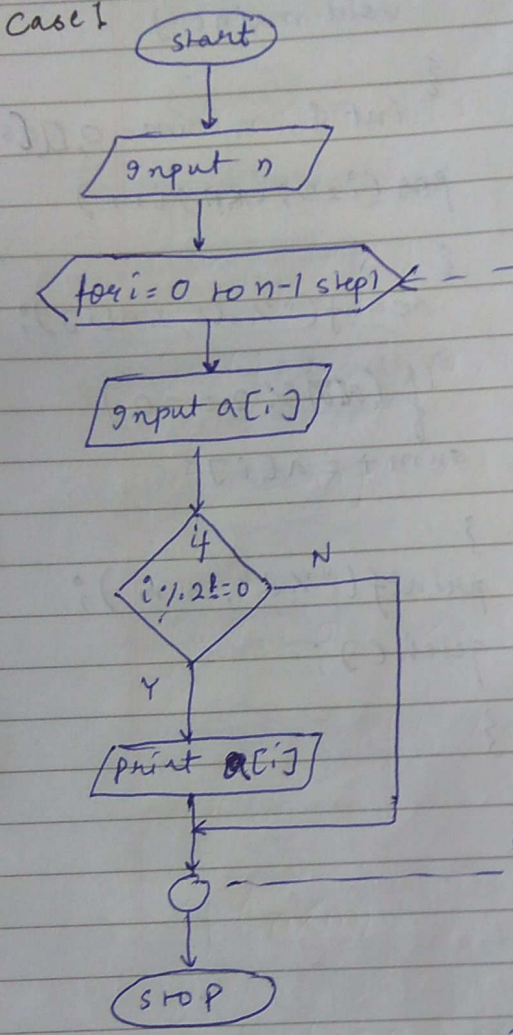

```
void main( )
{
  int i, n, sum = 0, [illegible]
for (i=0; i<n; i++)
{
  scanf("%d", &a[i]);
  if (a[i] % 2 == 0)
sum + = a[i];
}
printf("%d", sum );
getch( );
}
```

Smit is funny

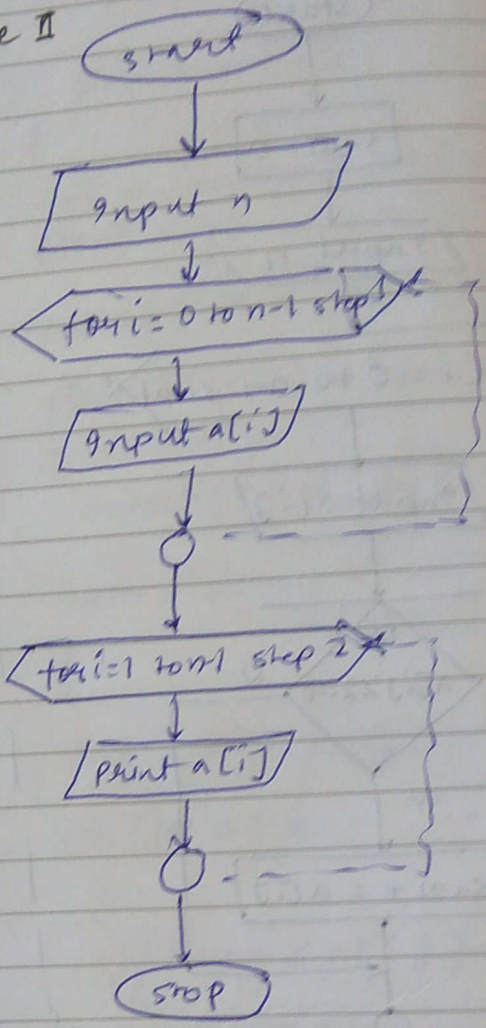
C = K & 3

Q. DAF to input n no. in an array and print all odd index value of that array

case I

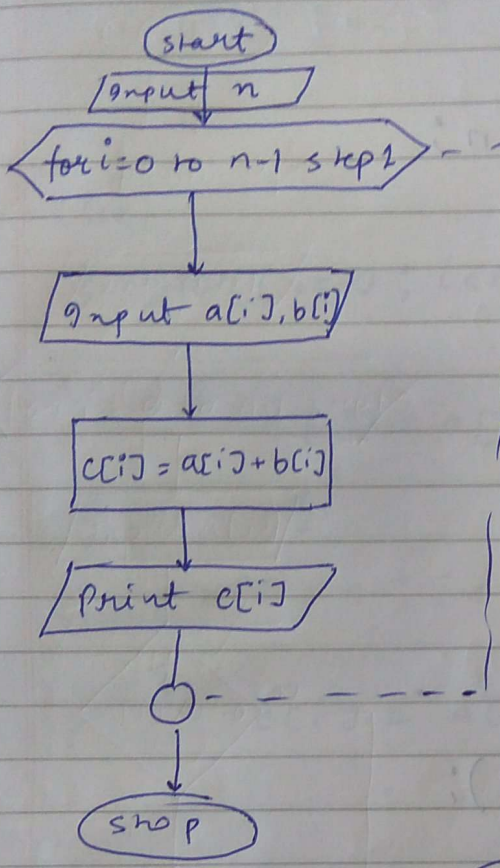


case II

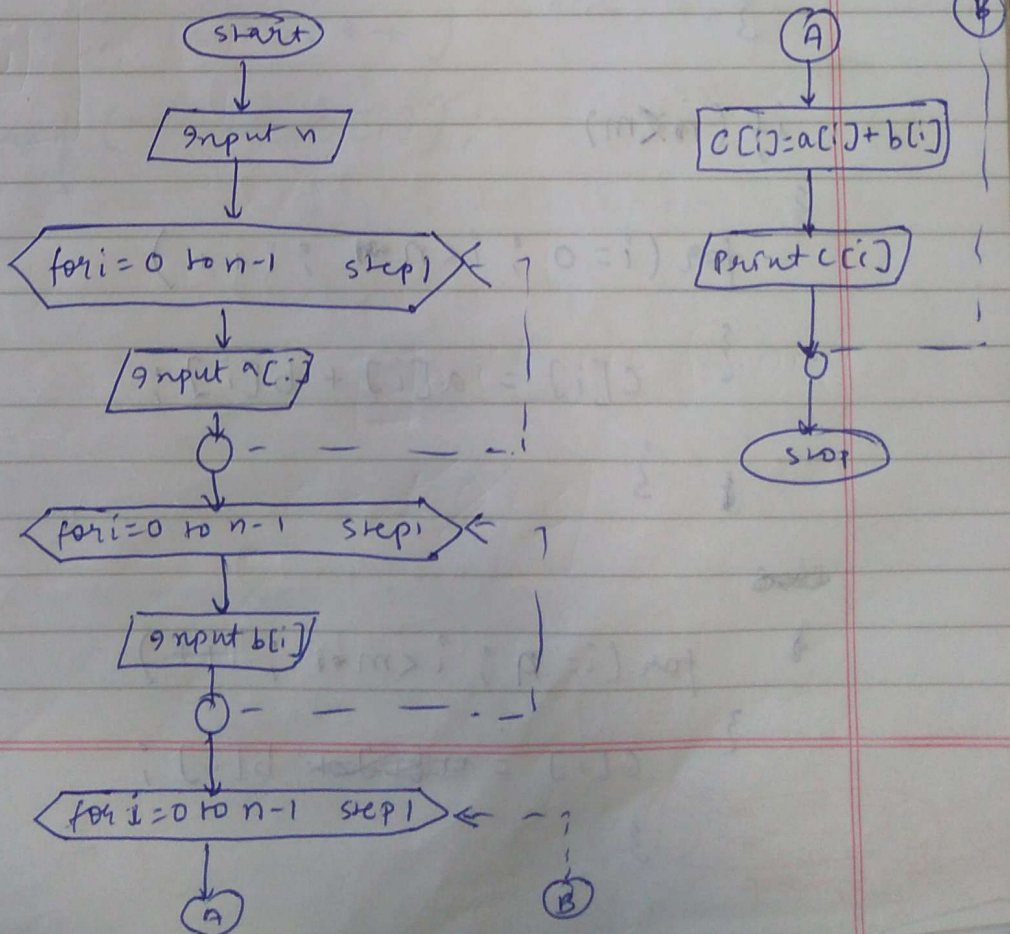



```
void main()
{ int i, n, a[50];
  scanf("%d", n);
  for(i=0; i<n; i++)
  { scanf("%d", &a[i]);
  }
  for(i=1; i<n; i+2)
  { printf("%d", a[i]);
  } getch(); }
```

Q. Pgm to input arrays (2) of equal size and print their sum storing in the 3rd array.




```
void main()
{
int a[100], b[100], c[100], i, n;
for(i=0; i<n; i++)
{
scanf("%d %d", &a[i], &b[i]);
c[i] = a[i] + b[i]
printf("%d", c[i])
}
getch();
}
```



Q. WAP to input 2 arrays of unequal size, add them and store them into 3rd array.

```
(1).  void main()
      {
        int a[10], b[10], i, m, n;
        scanf("%d", &n);
        scanf("%d", &m);
      for (i=0; i<m; i++)
        {  scanf("%d", &a[i]);
        }
      for (i=0; i<m; i++)
        {  scanf("%d", &b[i]);
        }
      if (n<m)
      {
        for (i=0; i<n; i++)
        {  c[i] = a[i] + b[i];
        }
      }
      {
        for (i=n; i<m; i++)
        {  c[i] = b[i];
        }
      }
```

```
for (i=0 ; i<m ; i++)
{
    printf("%d", c[i]);
}
}
else
{ for(i=0; i<m; i++)
    { c[i] = a[i] + b[i];
    }
for(i= m ; i<n ; i++)
{
    c[i] = a[i]
}
for(i=0 ; i<n ; i++)
{ printf("%d", c[i]);
}
}
getch()
}
```

(2).

```
void main()
{
int a[50] = {0}, b[50] = {0}, c[50], i, n, m;
scanf("%d %d", &n, &m);
for (i=0; i<n; i++)
  {
    scanf("%d", &a[i]);
  }
for (i=0; i<m; i++)
{
    scanf("%d", &b[i]);
  }
if (n<m)
{ for(i=0; i<m; i++)
  { c[i] = a[i] + b[i];
    print("%d", c[i]);
  }
else
{
  for(i=0, i<n; i++)
    c[i] = a[i] + b[i];
    printf("%d", c[i]);
}
getch();
}
```

Q. WAP to input a array and print it in reverse order.

```
void main()
{
    int a[10], i, n
    scanf("%d", &n);
    for(i=0; i<n; i++)
    {
        scanf("%d", &a[i]);
    }
    for(i=n-1; i>0; i--)
    {
        printf("%d", a[i]);
    }
    getch()
}
```

```
int n;
scanf("%d", &n);
int a[n];
```
(as per ub untu)

Q. WAP to input an array and print it in reverse order.

```
void main()
{   int n, temp;
    scanf("%d", &n);
    int a[n];
    for(i=0; i<n; i++)
    {   scanf("%d", &a[i]);
```

```
for (i=0; i<n/2; i++)
{ temp = a[i];
  a[i] = a[n-1-i];
  a[n-1-i] = temp;
}
for (i=0; i<n; i++)
{ printf("%d", a[i]);
{ getch();
}
```

Q. WAP to input n no. in an array and then find if a given no. is present in that array or not. If it is present then print its position otherwise print a message no. not present

```
void main()
{
  int n, i, num, c=0;
  scanf("%d", &n);
  int a[n];
  for(i=0; i<n; i++)
  { scanf("%d", &a[i]);
  }
```

Q.

```
scanf("%d", &num);
for(i=0 ; i<n ; i++)
{ if(num == a[i])
  { printf("position %d", i+1);
    c++;
  } }
    if(c==0)
      printf("not found");
getch();
}
```

Q. WAP to input 2 arrays and merge them into 3rd array.

```
void main()
{
  int n, m, c[40], i,
  scanf("%d", &n);
  int a[n];
for(i=0 ; i<n ; i++)
{ scanf("%d", &a[i]);
{
  scanf("%d", &m);
  int b[m];
for(i=0 ; i<m ; i++)
  { scanf("%d", &b[i]);
```

~~int c[n]~~

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

~~{~~

~~c[i] = a[i];~~

~~printf("%d", c[i]);~~

~~}~~

~~for (i=i+n; i<m+n;~~

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

{

c[i] = a[i];

}

for (i=0; i<m; i++)

{

c[i+n] = b[i];

}

c[i+n] = b[i]

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

{

printf("%d", c[i]);

}

getch()

}

Q. WAP to input n no. and store those no. in the array which are multiple ~~by~~ of 5.

```
void main ()
{
  int num, n, i, j = 0;
  for i=0
  scanf("%d", &n);
  int a[n];
for (i = 0; i<n; i++)
{
    scanf("%d", &num)
    if(num % 5 == 0)
    { a[j] = num;
      j++;
}
}
    for (i=0; i<j; i++)
    { printf("%d", a[i])
    }
}
```

Q. WAP to input n different no. in an array and then delete a given no. in that array.

```
void main()
{
  int n, i, num, j;
  scanf("%d", &n);
  int a[n];
  for(i=0; i<n; i++)
  { scanf("%d", &a[i]);
```

```
scanf("%d", &num);
for(i=0; i<n; i++)
{
    if(num==a[i])
    {for(j=i; j<n-1; j++)
        a[j] = a[j+1];
    }
}
```

Q. WAP a program to input n no. in an array then insert a no. in that array at a given index.

```
void main()
{
    int a[20], i, j, n, num, k
    scanf("%d", &n)
    for(i=0; i<n; i++)
    { scanf("%d", &a[i]);
    }
    scanf("%d", &num);
    scanf("%d", &k);
    for(i=0; j<n; j++)
    {
        if(i==k)
        {
            for(j=i; j<n+1; j++)
            for(i=n; i>k; i--)
            {
                a[i] = a[i-1]
            }
```

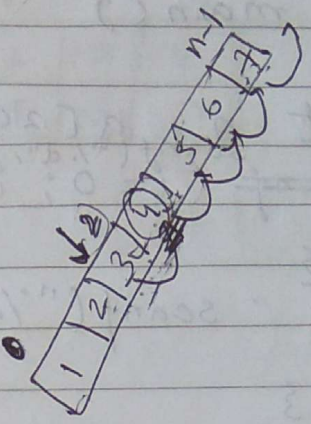
a[i]
n=6
i=6
a[5] = a[6]

```
for (i = n-1 ; i >= k ; i--)
{
    a[i+1] = a[i];
}
```

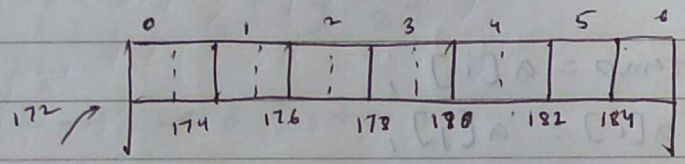
```
for (i = n ; i > k ; i--)
    a[i] = a[i-1];
```

```
a[k] = num;

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



Base Address + index * size of data type



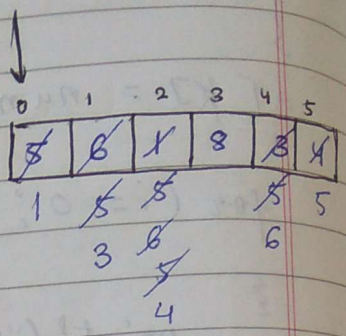
$= 172 + 3 \times 2$

$= 172 + 6$

$= 178$

Q. WAP to input n no. in an array and then sort them in ascending array without using another array.

```
void main()
{
   int n, a[20], i, j, temp;
   scanf("%d", &n);
   for (i=0; i<n; i++)
   {
      scanf("%d", &a[i]);
   }
   for(i=0; i<n; i++)
   {
      for(j=i+1; i<n; j++)
      {
         if(a[i] > a[j])
         {
            temp = a[i];
            a[i] = a[j];
            a[j] = temp;
         }
      }
   }
   for(i=0; i<n; i++)
   {
      printf("%d", a[i]);
   }
   getch();
}
```

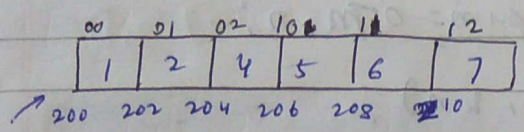


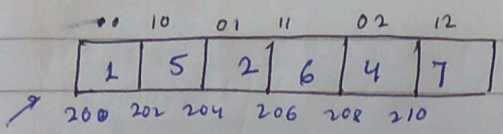
2D Array

datatype arrayname [row][column];

int a[2][3];

	0	1	2
0	1	2	4
1	5	6	7





Q. WAP to input a matrix of size 3×2 and print all elements.

```
void main()
{
  int i, j, a[3][2],
  for(i=0; i<3; i++)
  {
    for(j=0; j<2; j++)
    {
      scanf("%d", &a[i][j]);
    }
  }
for(i=0; i<3; i++)
{ for(j=0; j<2; j++)
  { printf("%d", a[i][j]);
    printf("\t");
  } printf("\n");
getch();
}
```

Q. WAP to input a matrix of size m x n and print the sum of all no.

```
void main()
{
   int a[10][10], i, j, sum=0, m, n;
   scanf("%d %d", &m, &n);
   for(i=0; i<m; i++)
   {
      for(j=0; j<n; j++)
      {
         scanf("%d", &a[i][j]);
      }
   }
   for(i=0; i<m; i++)
   {
      for(j=0; j<n; j++)
      {
         sum = sum + a[i][j];
      }
   }
   printf("%d", sum);
   getch();
}
```

Q. WAP to input a square matrix of size 4 and calculate sum of 2nd index column values.

```
void main()
{
  int a[4][4], i, j, sum=0;
  for(i=0; i<4; i++)
  {
    for(j=0; j<4; j++)
    {
      scanf("%d", &a[i][j]);
    }
  }
  for(i=0; i<4; i++)
  {
    for(j=0; j<4; j++)
    {
      if(j==2)
      {
        sum = sum + a[i][j];
      }
    }
  }
  printf("%d", sum);
  getch();
}
```

sum = sum + a[i][2];
(optimized)

Q. WAP to input two square matrix and perform matrix addition.

```
void main()
{
  int i, a[10][10], b[10][10], c[10][10], j, n
printf("enter order");
  scanf("%d", &n);
  for(i=0; i<n; i++)
  { for(j=0; j<n; j++)
    {scanf("%d", &a[i][j]);
    }
  }
  for(j=0; i<n; i++)
  {
    for(j=0; j<n; j++)
    {
      scanf("%d", &b[i][j]);
    }
  }
  for(i=0; i<n; i++)
  {
    for(j=0; j<n; j++)
    {
      c[i][j] = a[i][j] + b[i][j];
      printf("%d", c[i][j];
    }
  }
```

Scanned by CamScanner

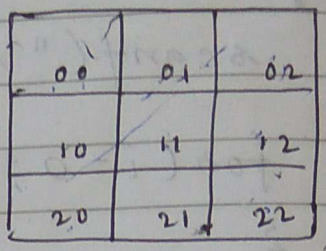
Q WAP to input a square matrix and print all principle diagonal element

```
void main()
{
  int i, j, a[10][10], n;
  {
    scanf("%d", &n);
    for(i=0; i<n; i++)
    {
      for(j=0; j<n; j++)
      {
        scanf("%d", &a[i][j]);
      }
    }
    for(i=0; i<n; i++)
    {
      for(j=0; j<n; j++)
      {
        if(i==j)
          printf("%d", &a[i][j]);
      }
    }
  getch();
}
```

```
for(i=0; i<n; i++)
{
  printf("%d", a[i][j]);
}
```

Q. WAP to input a square matrix of size n and print secondary diagonal elements.

```
void main()
{
  int i, j, n, a[10][10];
  scanf("%d", &n);
  for(i=0; i<n; i++)
  {
    for(j=0; j<n; j++)
    {
      scanf("%d", a[i][j]);
    }
```



n = 3

i+j

0+2

1+1

2+0

```
for(i=0; i<n; i++)
{
  for(j=0; j<n; j++)
  {
    if((i+j)==(n-1))
      printf("%d", a[i][j]);
  }
}
```

```
for(i=0; i<n; i++)
{
  printf("%d", a[i][n-1-i]);
}
```

```
for(i=0, j=n-1, i<n; i++, j--)
{
  printf("%d" a[i][j]);
}
```

Q. WAP to input a square matrix of size n and print upper triangle of principle diagonal.

```
void main()
{
  int i, j, n, a[10][10];
  scanf("%d", &n);
```

a[i][j]

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

```
for(i=0; i<n; i++)
{
 for(j=0; j<n; j++)
 {
   if(j>=i)
   {
     printf("%d", a[i][j]);
   }
 }
}
```

```
for(i=0; i<n-1; i++)
{
 for(j=i+1; j<n; j++)
 {
   printf("%d", a[i][j]);
 }
}
```

Q. WAP to input a square matrix and print lower triangle of principle diagonal

```
void main()
{
 int i, j, n, a[10][10];
 scanf("%d", &n);
 for(i=0; i<n; i++)
 {
  for(j=0; j<n; j++)
 {
  scanf("%d", a[i][j]);
 }
}
```

i=1 to (n-1) step 1
j=i-1 to (n-2) step

```
for(i=1; i<n; i++)
{
 for(j=i-1; j<n-1; j++)
```

```
for (i=0; i<n; i++)
{
  for (j=0; j<n; j++)
  { if (i>j)
  printf("%d", a[i][j]);
  }
}
}
```

Q. WAP to input a square matrix & print upper triangle of secondary diagonal

```
void main()
{
  int i, j, n, a[10][10];
  scanf("%d", &n);
  for (i=0; i<n; i++)
  {
    for (j=0; j<n; j++)
    {
      scanf("%d", a[i][j]);
    }
  }
  for (i=0; i<n; i++)
  {
    for (j=0; j<n; j++)
    {
      if (i+j < n-1)
      printf("%d", a[i][j]);
    }
  }
}
```

00	01	02
10	11	12
20	21	22

i = 0 ; i < n-1 ; i++
j = 0 ; j < n-1-i ; j++

```
for (i=0; i<n-1; i++)
{
for (j=0; j<n-1-i; j++)
{
printf("%d", a[i][j]);
}
}
```

Q. WAP to input a square matrix & lower triangle of secondary diagonal to print

```
void main()
{
  int i, j, n, a[10][10];
  scanf("%d", &n);
  for(i=0; i<n; i++)
  {
    for(j=0; j<n; j++)
    {
      scanf("%d", a[i][j]);
    }

  for(i=0; i<n; i++)
  {
  for(j=0; j<n; j++)
   {
     if(i+j > n-1)
     printf("%d", a[i][j]);
   }
  }
}
```

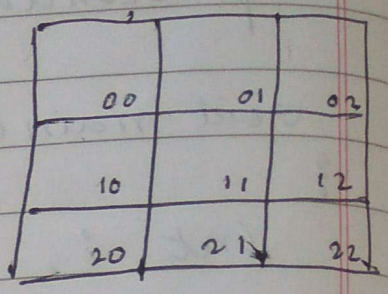
```
for(i=1; i<n; i++)
{
 for(j=n-1; j<n-1-i; j--)
 {
 printf("%d", a[i][j]);
 }
}
```

Q. WAP to input a square matrix and store and print its transpose in the same matrix

```
void main()
{
  int i, j, n, a[0][10], temp;
  scanf("%d", &n);
```

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

for (i=0 ; i<n ; i++)
{
  for (j=i+1; j<n ; j++)
   {
      temp = a[i][j];
      a[i][j] = a[j][i];
      a[j][i] = temp;
   }
for (i=0; i<n-1; i++)
{
  for (j=0; j<n ; j++)
   {
     printf("%d", a[i][j]);
   }
}
}
```



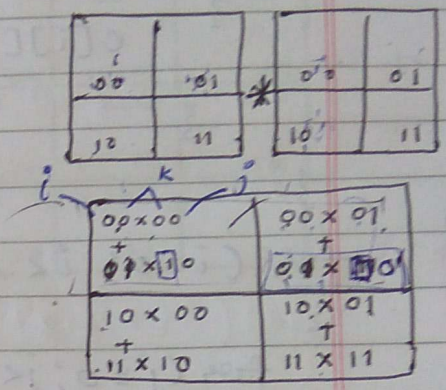
01
02

a[0][0]*b[0][0] + a[0][1]*b[1][0] + a[1][0]*b[0][1] + b[

Q. WAP to input two matrix and calculate matrix multiplication.

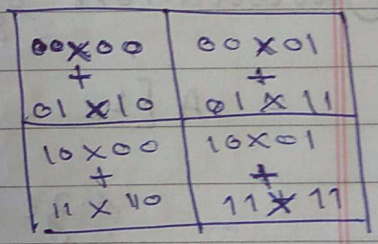
```
void main()
{
  int i, j, a[20][20], b[20][20], k, sum=0,
                    c[20][20]={0};
  scanf("enter");
  scanf("%d", &n);
  for(i=0; i<n; i++)
  {
  for(j=0; j<n; j++)
  {
     scanf("%d", a[i][j]);
  }
}
for(i=0; i<n; i++)
{
  for(j=0; j<n; j++)
  {
     scanf("%d", b[i][j]);
  }
}

  for(i=0; i<n; i++)
  {
   for(j=0; j<n; j++)
   { for(k=0; k<n; k++)
     {
        sum = sum + a[i][k] *b[k][j];
     }
  c[i][j] = sum;
  sum = 0;
```



1 2 * 2 3
3 4 3 4

1+6=7
7+6=13
13+6=19
19+6=25



```
if (c1 != r2)
    printf ("Multiplication not possible");
else
    for (i = 0; i < r1; i++)
    {
        for (j = 0; j < c2; j++)
        {
            for (k = 0; k < c1; k++)
            {
                c[i][j] = c[i][j] + a[i][k] * b[k][j];
            }
        }
    }
    for (i = 0; i < r1; i++)
    {
        for (j = 0; j < c2; j++)
        {
            printf ("%d", c[i][j]);
        }
    }
}
```

Calculation of Memory Address:

A [L_r . . . U_r] [L_c . . . . U_c]

size = w bytes

A [i] [j]

Base Address = BA

Row Major

Address (A[i][j]) = BA + w(m * (i - L_r) + (j - L_c))

Column Major

$$Address(A[i][j]) = BA + w*((i-L_R)+N*(j-L_C))$$

where; $M = U_C - L_C + 1$

$N = U_R - L_R + 1$

$A[0....2][0....2]$

200 (00)	202 (01)	204 (02)
206 (10)	208 (11)	210 (12)
212 (20)	214 (21)	216 (22)

$N = U_R - L_R + 1 = 2-0+1 \rightarrow 3$

$M = U_C - L_C + 1 = 2-0+1 \rightarrow 3$

$w = 2$ Bytes

$BA = 200$

$$Address(A[i][j]) = BA + w(M*(i-L_R)+(j-L_C))$$

$$= 200 + 2*(3\times(1-0)+(1-0))$$

$$= 200 + 2*(3+1)$$

$$= 200 + 2 + 4 = 208$$

200 (00)	206 (01)	212 (02)
202 (10)	208 (11)	214 (12)
204 (20)	210 (21)	216 (22)

$Address(A[1][2])$

$= BA + w*((i-L_R)+N(j-L_C))$

$= 200 + 2*((1-0)+3*(2-0))$

$= 200 + 2*(1+6)$

$= 214$

$A[\overset{L_R}{-10} \ldots \overset{U_R}{10}][\overset{L_C}{10} \ldots \overset{U_C}{40}]$

BA = 300

w = 4 Bytes

A[2][12]

M = 40 − 10 + 1 = 31

N → 10 − (−10) + 1 = 21

Add(A[2][12]) = 300 + 4 * ((2 − (−10)) * 31 + (12 − 10)

= 300 + 4 * (372 + 2)

= 300 + 4 * 374

= 300 + 1496 = 1796