LAB-ASSIGNMENT 3 (Array and Pointer)

SUBJECT: Computer Lab.   SUBJECT CODE: CS111. SEMESTER: I

Linear
search

H.W
H.W

f.w

1. Write a C program to take N numbers as an input in an array and print them.
2. Write a C program to insert a number in a given position in an array.
3. Write a C program to delete a number in a given position in an array.
4. Write a C program to search a number in an array and also print the position of the input number.
5. Write a C program to sort an array element. (insertion, bubble, selection)
6. Write a C program to print the address of a given input.
7. Write a C program to count the number of vowel and consonant in a character array.
8. Write a C program to take M X N matrix as an input and print the matrix properly.
9. Write a C program to addition and subtraction of two matrices.
10. Write a C program to transpose of a matrix.
11. Write a C program to multiplication of two matrices.
12. Write a C program to addition and subtraction of two matrices (using pointer and malloc() function).

$f(a[i] = "a" | "e" | "i" | "o" | "u")$ scanf ("·/-p" & a)

```c
#include <stdio.h>

int main()
{
    int a[1000],i,n;

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

    printf("Enter %d elements in the array : ", n);
    for(i=0;i<n;i++)
    {
        scanf("%d", &a[i]);
    }

    printf("\nElements in array are: ");
    for(i=0;i<n;i++)

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

    return 0;
}
```

a[100]

0	1	2	3	4	
5	10	2	30	40	

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

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

Insert an element at a specific position in an Array

insert

50

x

pos = 5

1	2	3	4	5	6	7 ·	8 ·	9 ·	10 ·

Initial Array

1	2	3	4	50	5	6	7	8	9	10

Array with X inserted at position pos
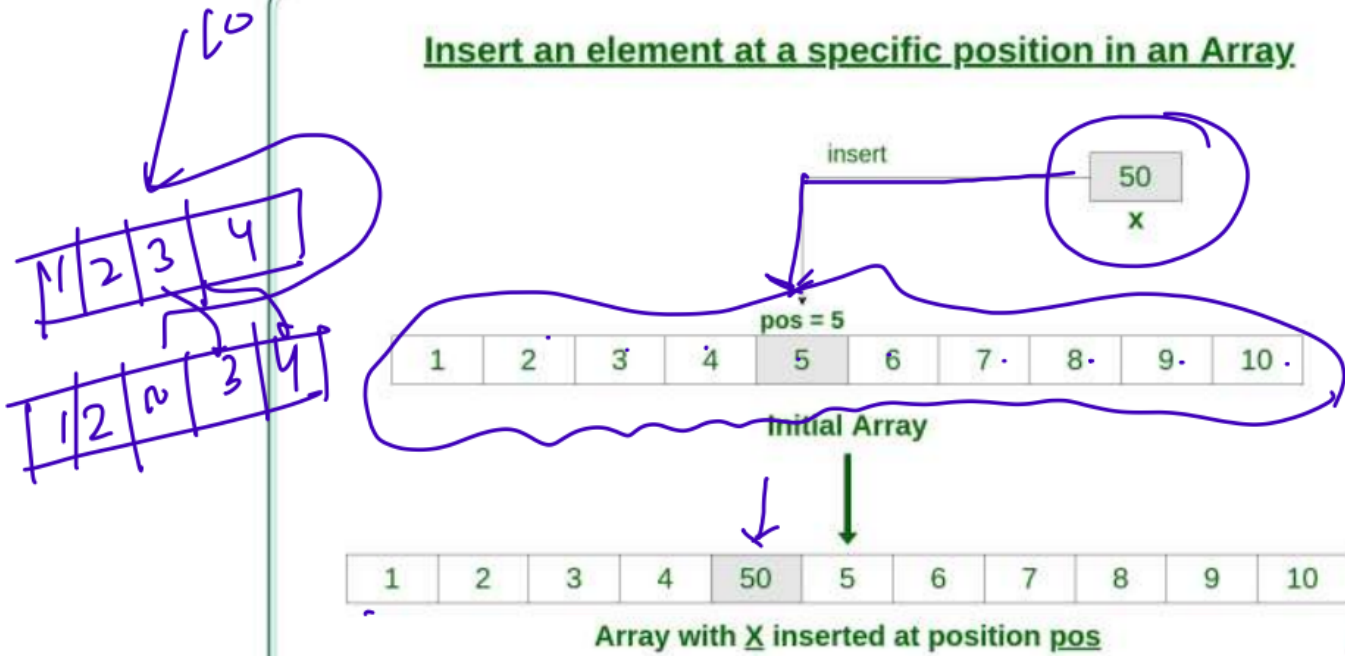
𝒢G

1. First get the element to be inserted, say x
2. Then get the position at which this element is to be inserted, say pos
3. Then shift the array elements from this position to one position forward, and do this for all the other elements next to pos.
4. Insert the element x now at the position pos, as this is now empty.

```c
// C Program to Insert an element
// at a specific position in an Array

#include <stdio.h>

int main()
{
        int arr[100] = { 0 };
        int i, x, pos, n = 10;

        // initial array of size 10
        for (i = 0; i < 10; i++)
                arr[i] = i + 1;

        // print the original array
        for (i = 0; i < n; i++)
                printf("%d ", arr[i]);
        printf("\n");

        // element to be inserted
        x = 50;

        // position at which element
        // is to be inserted
        pos = 5;

        // increase the size by 1
        n++;

        // shift elements forward
        for (i = n-1; i >= pos; i--)
                arr[i] = arr[i - 1];

        // insert x at pos
        arr[pos - 1] = x;

        // print the updated array
```
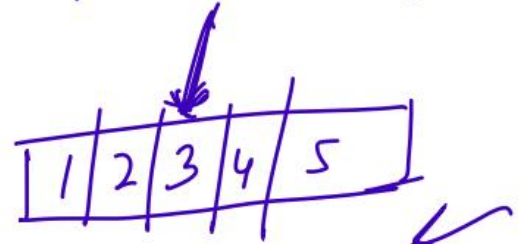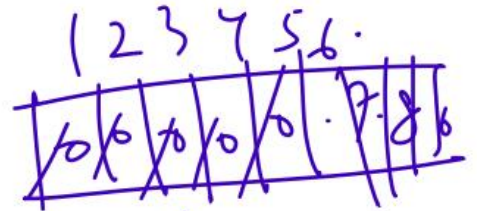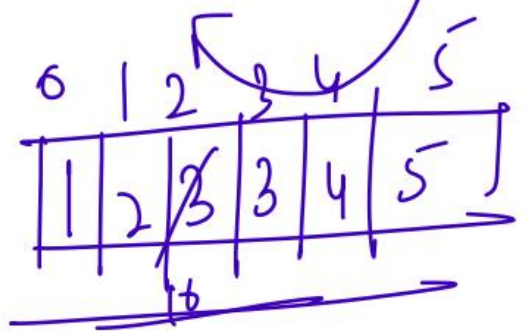
$$a[i] = a[i-1]$$

$$a[pos-1] = x$$

$$a[3-1] =$$
$$a[2] = 50 \; 10$$

```c
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");

    return 0;
}
```

Delete element from an array

Array

| 5 | -3 | 7 | 6 | 2 | 1 | 8 | 3 | 0 | 4 |

10 elements (size = 10)

Array after deleting element at index 6

| 5 | -3 | 7 | 6 | 2 | 1 | 3 | 0 | 4 | 4 |

9 elements (size = 10)

programmingsimplified.com

C program to delete an element in an array: This program deletes or removes an element from an array. A user will enter the position at which the array element deletion is required. Deleting an element does not affect the size of the array. It also checks whether deletion is possible or not, for example, if an array contains five elements and user wants to delete the element at the sixth position, it isn't possible.

Remove element from array C program

```c
#include <stdio.h>
int main()
{
   int array[100], position, c, n;
```

```c
    printf("Enter number of elements in array\n");
    scanf("%d", &n);

    printf("Enter %d elements\n", n);

    for (c = 0; c < n; c++)
      scanf("%d", &array[c]);

    printf("Enter the location where you wish to delete element\n");
    scanf("%d", &position);

    if (position >= n+1)
      printf("Deletion not possible.\n");
    else
    {
      for (c = position - 1; c < n - 1; c++)
        array[c] = array[c+1];

      printf("Resultant array:\n");

      for (c = 0; c < n - 1; c++)
        printf("%d\n", array[c]);
    }

    return 0;
}



/* Linear search in an array*/
```

```c
//LINEAR SEARCH
#include <stdio.h>
#define MAX 50
int main(void)
{
    int i=0,n,item,arr[MAX];
    printf("Enter the number of elements : ");
    scanf("%d",&n);
    printf("Enter the elements : \n");
    for(i=0; i<n; i++)
        scanf("%d", &arr[i]);
    printf("Enter the item to be searched : ");
    scanf("%d", &item);
    for(i=0;i<n;i++)
      if(arr[i]==item)
      {
    printf("%d is present at location %d.\n", item, i+1);
    break;
    }

  if (i == n)
    printf("%d isn't present in the array.\n", item);

    return 0;
    }
```
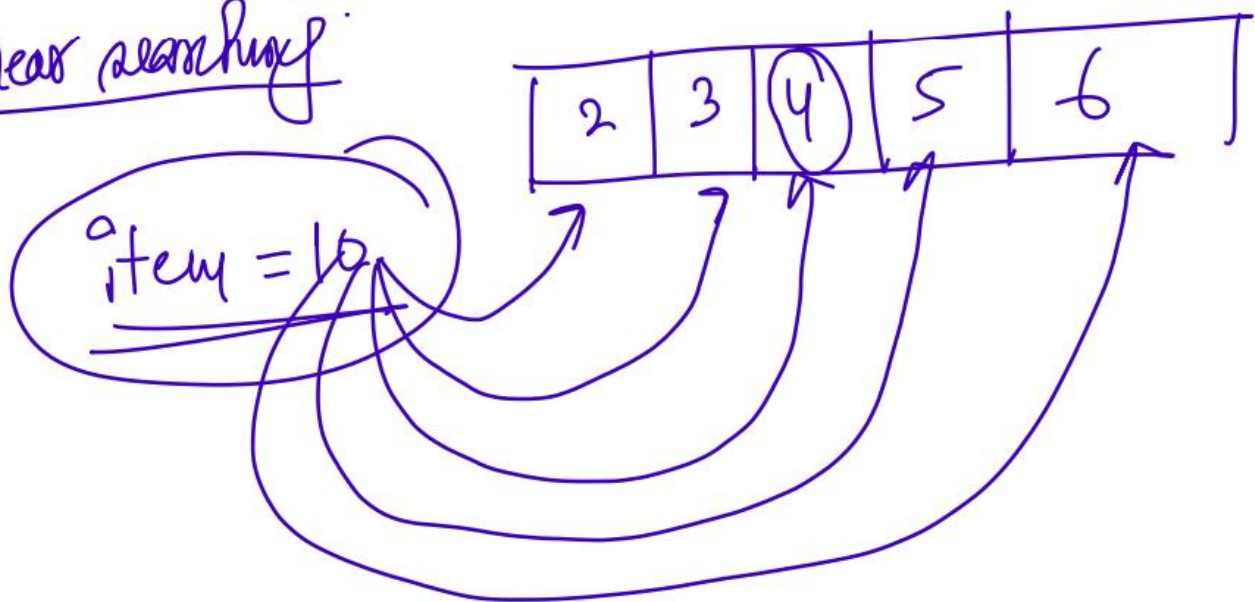
worst Case

$$O(n)$$

Linear searching

item = 10

| | 2 | 3 | 4 | 5 | 6 | |

```c
//BINARY SEARCH

#include <stdio.h>

int main()

{

int i, low, high, mid, n, key, array[100];

printf("Enter number of elements……..");

scanf("%d",&n);

printf("Enter %d integers", n);

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

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

printf("Enter value to find____");

scanf("%d", &key);

low = 0;

high = n - 1;

mid = (low+high)/2;

while (low <= high) {

if(array[mid] < key)

low = mid + 1;

else if (array[mid] == key) {

printf("%d found at location %d", key, mid+1);

break;

}

else

high = mid - 1;

mid = (low + high)/2;

}

if(low > high)
```

```c
    printf("Not found! %d isn't present in the list", key);

    return 0;

}
```

2D Array

int a[3][2] = {{1, 2, 3}, {4, 5, 6}}   { {1, 2},
                                                  {3, 4},
no of rows   no of columns   0 first row   1st col   {5, 6}
                                                       };

0th row →

1 a_{00}	2 a_{01}
3 a_{10}	4 a_{11}
5 a_{20}	6 a_{21}

1st row →
2nd row →

int a[5] = {10, 20, 30, 40, 50};

10	20	30	40	50

a[0][0]	a[1][0]
a[0][1]	a[1][1]
a[0][2]	a[1][2]

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

a_{00} a_{01} a_{02}
a_{10} a_{11} a_{12}
a_{20} a_{21} a_{22}

row major order

a_{01} → 104

a_{00}	a_{01}	a_{02}	a_{10}	a_{11}	a_{12}	a_{20}	a_{21}	a_{22}
100	104	108	112	116	120	124	128	

col = 112   column major order

100	104	108	112	116	120	124		
a_{00}	a_{10}	a_{20}	a_{01}	a_{11}	a_{21}	a_{02}	a_{12}	a_{22}

$\theta(n^2)$

addition

```
for (i=0; i<N; i++)
  for (j=0; j<M; j++)
    print (" %d", a[i][j]);
```

$a[\ ]$ $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ $b[\ ]$ $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

$c = a[\ ] + b[\ ] = \begin{bmatrix} 2 & 4 & 6 \end{bmatrix}$

```
take    1st Matrix   a[ ][ ]
take    2nd Matrix   b[ ][ ]

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

            c[i][j] = a[i][j] (+) b[i][j];
    }

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

$$a = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \\ a_{20} & a_{21} \end{bmatrix}$$

$$a^T = \begin{bmatrix} a_{00} & a_{10} & a_{20} \\ a_{01} & a_{11} & a_{21} \end{bmatrix}$$

$$a = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

$$a^T = \begin{bmatrix} 1 & 5 & 9 \\ 2 & 6 & 10 \\ 3 & 7 & 11 \\ 4 & 8 & 12 \end{bmatrix}$$

Matrix Multiplication

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} * \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 1*1 + 2*3 & 1*2 + 2*4 \\ 3*1 + 4*3 & 3*2 + 4*4 \end{bmatrix}$$

$2 \times 2 \qquad 2 \times 2$

$$= \begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix}$$

$O(n^2)$

$O(n^3)$

Strings.

array of type char

'A'

"debayan"

| d | e | b | a | y | a | n | \0 |

debayan | ganguly

gets (str)
puts (str)

buffer :- A temporary storage area.

buffer
| | 4 | 1 |

Sender | 4 | 3 | 2 | 1 | ... receiver

gets (str);

| d | e | b | a | y | a | n | y | a | c | h | g | u | l | y | \0 | | | | | |
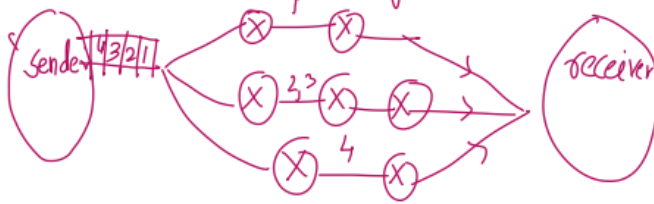
strlen()
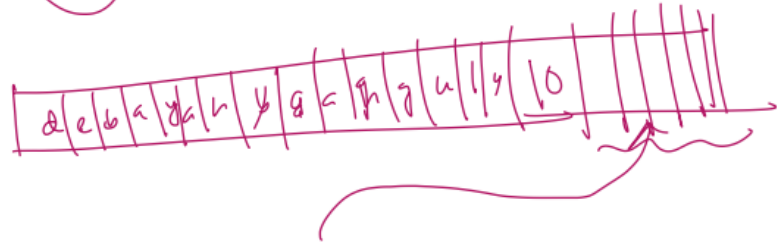
strcpy()
strupr()
strrev()

debayan_ganguly/o   O/p :- 15

* __Multiplication__

* implement __strlen()__ without using __library function__

```
{  while ( str[i] != '\0' )    //x
        count ++;
```