* Program to swap two numbers using call by reference.

```
#include <stdio.h>
void accept (int * a)
{    printf ("enter the value\n");
     scanf ("%d", &a)
}
Void swap (int * a , int * b)
{    int t ;
     t  = * a;
     *a  = *b;
     *b  = t;
}

int main()
{    int a,b;
     pf (" enter the value of a \n");
     accept (&a);
     pf (" enter the value of b \n");
     accept (&b);
     swap (&a, &b);
     pf (" swaped values are %d  %d \n", a, b);
}
```

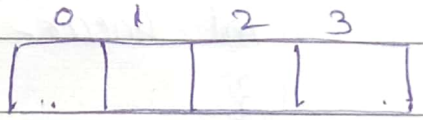
```
void swap
(int* x,int*y )
{  int temp;
   temp = *x;
   *x = *y;
   *y = temp;
}
int main()
{  int a = 10,
       b = 20
   swap (&a, &b)
   pf("a = %d,
       b = %d",
       a, b);
}
```

(2) program to add 2 numbers

```
#include<stdio.h>
void sum_func (int x, int y)
{       int sum;
        sum = x+y;
        return sum;
}
int main()
{       int a,b;
        int result;
        result = sum_func(a,b);
        printf("sum = %d \n", result);
        return 0;
}
```

DATE : / /

③ program to accept the elements and display the elements of array (without using functions)




```
#include <stdio.h>
int main()
{   int a[10], i, n=4;
    printf("enter the numbers \n");
    for (i=0 ; i<n ; i++)
        scanf("%d", &a[i]);
    printf("array elements are \n");
    for (i=0 ; i<n ; i++)
        printf("%d \n", a[i]);
}
```

④ accept & display (using functions)

```
void accept (int *b)
{   int i;
    pf("enter elements\n");
    for(i=0; i<n; i++)
        scanf("%d", &b[i]);
}

int main()
{   int a[4];
    accept(a);
    display(a);
}
```

```
void display (int *c)
{   int i;
    pf("elements are \n");
    for(i=0; i<n; i++)
        pf("%d \n", c[i]);
}
```

or

```
int main()
{   int a[4], n=4;
    accept(a,n);
    display(a,n);
}
void accept (int *b, int n)
```

5 linear search

```
#include <stdio.h>
int linear_search (int a[], int search, int n)
{
    for (int i=0 ; i<n ; i++)
    {       if (a[i] == search)
            {   pf("element found in position %d \n", i+1);
                return 0;
            }
    }
    printf ("element not found \n");
}
int main()
{   int i, a[20], n, search, position;
    pf ("enter the no. of elements \n");
    sf ("%d", &n);
    pf ("enter the elements \n");
    for (i=0 ; i<n ; i++)
    {   scanf ("%d", &a[i]);
    }
    printf ("enter the no to be searched \n");
    sf ("%d", & search);
    position = linear-search (a, search, n);
}
```

3/oct

DATE : / /

1. program to implement strrev without using inbuilt function.

```
#include <stdio.h>
#include <string.h>
void reverse (char s[])
{   int len ,i ;
    len = strlen(s);
    for (i =len -1 ; i>=0 ; i++)
    {
        printf("%c", s[i]);
    }
}
int main ()
{   char s1[20];
    printf(" enter a string : \n" );
    scanf ("%s", s1);
    reverse (s1);
    return 0;
}
```

(2)

```
#include <stdio.h>
#include <stdlib.h>
 int main()
{   int a[10], n, i;
    system ("cls");
    printf("Enter the number to convert : ");
    scanf("%d", &n);
    for (i=0; n>0; i++)
    {   a[i] = n%2;
        n = n/2 ;
    }
    printf("\n In Binary of Given Number is =");
    for (i = i-1 ; i>=0; i--)
    {  printf("%d", a[i]);
    }
    return 0;
}
```

Output

Enter the number to convert : 45.6

Binary of Given number is = 101101

③ Greatest of 3 Numbers

```
#include <stdio.h>

int largestNumber (int a, int b, int c)
{   int largest = 0;
    if (a>b && a>c)
        largest = a;
    else if (b>=a && b>c)
        largest = b;
    else
        largest = c;
    return largest;
}
int main ()
{   int a, b, c;
    printf ("Enter three numbers:");
    scanf("%d %d %d", &a, &b, &c);
    printf ("Largest number is : %d \n", largestNumber(a,b,c
    return 0;
}
```

Output:

```
Enter three numbers : 45
  58
  79
Largest number is : 79
```

49 MATRIX Addition

```
#include <stdio.h>
void accept (int *a, int *b, int m, int n)
{
   int i;
   printf ("Enter elements of 1st matrix \n");
   for (i=0; i<m; i++)
    {
        scanf ("%d", &a[i]);
    }
   printf ("enter elements of 2nd matrix \n");
   for (i=0; i<n; i++)
    {
        scanf ("%d", &b[i]);
    }
}
void add (int *a, int *b, int *c, int n)
{
    int i;
    for (i=0; i<n; i++)
    {
        c[i] = a[i] + b[i];
    }
    printf ("Resultant array \n");
    for (i=0; i<n; i++)
    {   printf ("%d \t", c[i]);
    }
}
```

```
int main()
{ int a[20], b[20], c[20], m, n;
  printf("enter number of elements in 1st matrix \n");
  scanf("%d", &m);
  printf("enter number of elements in 2nd matrix \n");
  scanf("%d", &n);
  accept(a, b, m, n);
  add(a, b, c, n);
  return 0;
}
```

5> Reverse a number

```
#include <stdio.h>
#include <stdlib.h>
void reverse(int n)
{ if (n == 0)
    printf("cannot be reversed \n");
  while (n != 0)
  { int digit;
    digit = n % 10;
    n = n / 10;
    printf("%d", digit);
  }
}
int main()
{ int n;
  printf("enter a number\n");
  scanf("%d", &n);
  reverse(n);
  return 0;
}
```

Sankpal Building, Behind Syndicate Bank, Main Road, RAIBAG-591 317. Ph./Fax : 08331-224224
Cell : 8088592666, 9916255170, e-mail : nictrbg@gmail.com, website : www.nictraibag.com

Program to accept, insert & delete elements of array.

```
#include <stdio.h>
int dele_pos (int *a, int n)
{
    int init_start, pos_term, elem, pos;
    printf("enter the position to be deleted \n");
    scanf("%d", &pos);
    if (pos >= 0 && pos < n)
    {
        init_start = pos;
        pos_term = n;
        elem = a[pos];
        printf("element deleted is %d \n", elem);
        while (1)
        {
            a[init_start] = a[init_start + 1];
            init_start++;
            if (init_start == pos_term - 1)
                break;
        }
        n--;
    }
    else
        printf("enter the position value
                within the range \n");
    return n;
}
```

```
int display(int *b, int *a)
{
    int i;
    printf("elements of array \n");
    for(i=0; i<*a; i++)
        printf("%d \n", b[i]);
}

void accept(int a, int *n)
{
    int i;
    printf("enter the elements \n");
    for(i=0; i<a; i++)
        scanf("%d", &n[i]);
}

int inser_position(int *a, int n)
{   int pos, elem;
    printf("enter the pos and element to be
                inserted \n");
    scanf("%d%d", &pos, &elem);
    int init_start = n, pos_term = pos;

    if(pos>=0 && pos<n)
    {
        while(i)
        {  a[init-start] = a[init-start-1]; //a(3)=a[2]
           init-start--;    //init-start=2
```

```
        if (init-start == pos-term)
            break;
    }
    a[pos] = elem;
    (n)++;
}
else
    printf("enter the position within
            the range \n");

return n;
}

int main()
{   int n=4, a[20] = {22,33,55,66};
    printf("enter the number of elements \n");
    scanf("%d", &n);                    // 4
    accept(n, a);                       // 4   2000
    display(a, &n);                     // a   2000
    // binsearch(a,n);
    // linsearch(a,n);
    n = inser-position(a,n);
    display(a, &n);
    n = dele-pos(a,n);
    display(a, &n);
}
```

Output :-

enter the number of elements
5
enter the elements
22
33
44
55
66
elements of array
22 33 44 55 66
enter the pos and element to be inserted
2
88
elements of array
22
33
88
44
55
66
enter the position to be deleted
2
element deleted is 88
elements of array
22
33
44
55
66

H.W

NICT NATIONAL INSTITUTE OF COMPUTER TECHNOLOGY COMPUTER TRAINING CENTRE

Page No.
Date : / /

① Program to perform Array Multiplicatn

```
#include <stdio.h>
#include <math.h>
void accept_arr1 (int m, int n, int a[10][10])
{
    int i, j;
    printf ("enter the array 1 elements \n");
    for (i=0; i<m; j=i+1)
    {
        for (j=0; j<n; j=j+1)
        {
            scanf ("%d", &a[i][j]);
        }
    }
    return;
}
void display (int m, int s, int c[10][10])
{
    int i, j;
    printf ("Result \n");
    for (i=0; i<m; i++)
    {
        for (j=0; j<s; j++)
        {
            printf ("%d", c[i][j]);
        }
        printf ("\n");
    }
    return;
```

```
void accept_arr2 (int r, int s, int b[10][10])
{
    int i, j;
    printf ("enter the array 2 elements \n");
    for (i=0; i<r; i++)
    {
        for (j=0; j<s; j++)
        {
            scanf ("%d", &b[i][j]);
        }
    }
    return;
}
void multiply (int m, int s, int n, int a[10][10],
               int b[10][10], int c[10][10])
{
    int i, j, k;
    for (i=0; i<m; i++)
    {   for (j=0; j<s; j++)
        {
            c[i][j] = 0;
            for (k=0; k<n; k++)
            {
                c[i][j] += a[i][k] * b[k][j];
            }
        }
    }
    return;
}
```

```
int main()
{
    int a[10][10], b[10][10], c[10][10]
    int m, n, r, s, i, j;
    printf("enter the Row and column of
                    array 1 \n");
    scanf("%d %d", &m, &n);
    accept - arr1(m, n, a);
    printf("enter the Row and Column of
                    array 2 \n");
    scanf("%d %d", &r, &s);
    accept - arr2(r, s, b);
    if (n == r)
        {
            multiply(m, s, n, a, b, c);
        }
    else
        printf("Multiplication of given matrix
                    is not possible \n");
    display(m, s, c);
    return 0;
}
```

2 Program to remove duplicate elements in the array.

```
#include <stdio.h>
void display (int *a, int n)
{
    printf("your new array is \n");
    for (int i=0; i<n; i++)
    {   printf ("%d", a[i]);
    }
}
void accept (int *a, int n)
{   printf ("enter the elements of array \n");
    for (int i=0; i<n; i++)
    {   scanf ("%d", &a[i]);
    }
}
void duplicate (int *a, int *b, int n)
{   for (int i=0; i<n; i++)
    {   for (int j=i+1; j<n; j++)
        {
            if (a[i]==a[j])
            {   for (int k=j; k<n; k++)
                {   a[k] = a[k+1];
                }
                n--;
                j--;
            }
        }
    }
}
```

```
    display (a, n);
}
int main()
{   int a[100], b[100], i, j, n;
    printf ("enter the length of array \n");
    scanf ("%d", &n);
    accept (a, n);
    duplicate (a, b, n);
    return 0;
}
```

Output

```
Enter length of array
4
enter elements of array
2
2
3
5
Your new array is
2
3
5
```

(3) Menu driven program to accept array elements, ~~delete~~ elements in specific posith, delete elements from specific posith

accept

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
void display (int a[], int n)
{   int i;
    printf("your array is \n");
    for (i=0; i<n; i++)
        printf ("%d", a[i]);
}
void accept-1 (int a[], int n)
{   int i;
    printf (" enter the array elements \n");
    for (i=0; i<n; i++)
        scanf ("%d", &a[i]);
    display (a, n);
}
void accept (int a[], int n)
{   int i;
    printf ("enter the array elements \n");
    for (i=0; i<n; i++)
        scanf ("%d", &a[i]);
}
void display-1 (int a[], int n)
{   int i;
    accept (a, n);
    printf ("your array is");
    for (i=0; i<n; i++)
```

```
        printf ("%d ", a[i]);
}
void add (int a[], int n)
{       int init_start, pos_term, i, pos, elem;
        accept (a, n);
    printf ("Enter the ~~element~~ postion of element
                to be inserted \n");
    scanf ("%d", &pos);
    printf ("Enter the element to be inserted \n");
    scanf ("%d", &elem);
    if (pos >= 0 && pos < n)
    {       init_start = n;
            pos_term = pos;
            while (1)
            {   a[init_start] = a[init_start-1];
                init_start--;
                if (init_start == pos)
                    break;
            }
            a[pos] = elem;
            n++
            display (a, n);
    }
    else
    {   printf("enter the position within range \n");
    }
}
```

```
void delete (int a[], int n)
{   int init_start, pos_term, i, pos, elem;
    accept (a, n);
    printf ("enter the position of element to
                be deleted \n");
    scanf ("%d", &pos);
    if (pos > 0 && pos < n)
    {   init_start = pos;
        pos_term = n;
        while (1)
        {   a[init_start] = a[init_start + 1];
            init_start ++;
            if (init_start == pos_term - 1)
                break;
        }
        elem = a[pos];
        n--;
        printf ("your new array is \n");
        display (a, n);
    }
    else
    {   printf ("enter the position within range \n");
    }
}
void select (int a[], int n, int p)
{   switch (p)
    {   case 1: accept_1(a, n);
                break;
```

```
        case 2 : add(a,n);
                 break;
        case 3: delete (a,n);
                 break;
        case 4 : display-1(a,n);
                 break;
        default : printf("enter the value within
                          range \n");
                 break;
    }
}
int main()
{
    int a[100], i, p, n, q;
    printf("enter the length of array \n");
    scanf("%d", &n);
    printf("enter the operation to be performed\n
            1. Accept the array \n 2. Accept
            elements in specific position \n
            3. Delete elements from specific
            position \n. 4 Display numbers \n");
    scanf("%d", &p);
    select(a, n, p);
    return 0;
}
```

4. Program to find largest, second largest & first smallest & second smallest element in array.

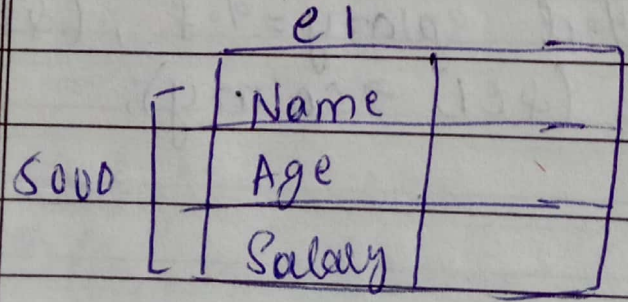
```
#include <stdio.h>
void display (int *a , int n)
{   for (int i=n-1; i>=0; i--)
    {
        printf("Largest element is = %d \n", a[i]);
        printf("second largest element is = %d \n",
                                    a[i-1]);
        break;
    }
    for (int i=0; i<n; i++)
    {
        printf("second smallest element is = %d \n",
                                    a[i+1]);
        printf("Smallest element is = %d \n", a[i]);
        break;
    }
}
void short_array (int *a , int * n)
{   int max=0, i, j;
    for (i=0; i<*n; ++i)
    {   for (j=i+1; j<*n; ++j)
        {   if (a[i] > a[j])
            {   max = a[i];
                a[i] = a[j];
                a[j] = max;
            }
        }
    }
}
```

```
void accept (int *a, int *n)
{
    printf ("enter the elements of array \n");
    int i;
    for (i=0; i<*n; i++)
        scanf ("%d", &a[i]);
}
int main()
{
    int a[100], n;
    printf ("enter the length of array \n");
    scanf ("%d", &n);
    accept (a, &n);
    short_array (a, &n);
    display (a, n);
    return 0;
}
```

```
#include <stdio.h>
typedef struct emp
{
    char name[20];
    int age;
    float salary;
}E;
int main()
{   E e1;
    scanf("%s %d %f", e1.name, &(e1.age),
                      &(e1.salary));
    printf("name = %s age = %d salary = %f\n",
            (e1).name, (e1).age, (e1).salary);
}
```

input → ach 18 5000

output → name = ach age = 18 salary = 5000.



Ipnput → Kis 42 60000

```
#include <stdio.h>
typedef struct emp
{       char name[20];
        int age;
        float salary;
} E;
void accept (E *e1)
{   printf("Enter name, age and salary\n");
    scanf("%s%d%f", (*e1).name, &(*e1).age,
                    &(*e1).salary);
}
void display (E e1)
{   printf("name = %s  age = %d  salary = %f",
            (e1).name, (e1).age, (e1).salary)
}
int main ()
{   E e1;
    accept (&e1);
    display (e1);
}
```

Output box

```
Enter name, age & salary
ach 22 50000
name = ach age = 22 salary = 50000.
```

glocal

* Pattern matching algorithm finds a given pattern P in a string text T. P length is assumed not to exceed the length of T

Algorithm

1. [initialize] set K = 1 and MAX = S - R + 1
2. Repeat steps 3 to 5 while K <= MAX
3. Repeat for L = 1 to R [test each char of P]
   If P[L] != T[K + L - 1] then go to step 5
   [end of inner loop]
4. [success] set INDEX = K and EXIT
5. Set K = K + 1 //sliding window to next comperison
   [end of step 2 outer loops]
6. [failure] set INDEX = 0
7. exit

P and T are string with lengths R & S, respectively and are stored as arrays. This algorithm finds the index [INDEX] of P in T

window :- pattern (P) is compared with main string (T) is called window

MAX = 2 means we can make two windows.

Program to pattern match.

```
#include <stdio.h>
#include <string.h>
int pattern_match (char P[20], char T[20])
{
    int R, S, MAX, INDEX, L, K;
    R = strlen(p);   // length of pattern
    S = strlen(T);   // main string
    INDEX = -1;   MAX = S-R+1;
    for (K=0; K<=MAX; K++) //window K=0
    {          //comparing two window.
        for (L=0; L<R; L++) //start from
        {.     if (P[L] != T[K+L]      beg of the
                   break;
        }
        if (L==R)  //a==i  L=0  R=3
            return K;   //sucess
        }
        return -1;
    }
    return -1;          // failure
}
int main()
{     char p[] = "ind", t[] = "aind";
    printf("pattern position = %d", pattern_match(p,t));
     return 0;
}
```

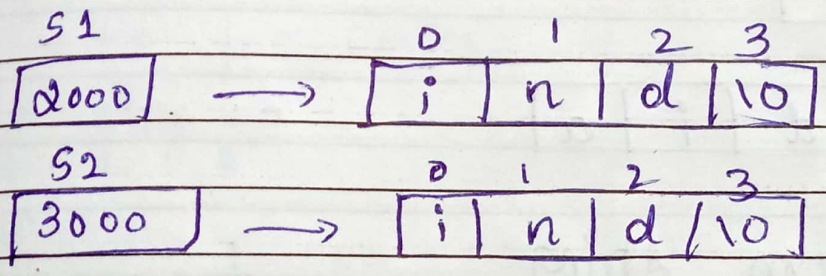
String length check function in <cstring>

```
// string prototype, already included in string.h
// returns length of string (not counting '\0').
int strlen(const char []);

int string-length = strlen("abcde");
// string-length is set to 5

int str-len(char str[20])
{
    int i=0;
    while(str[i]!='\0')
    {
        i++;
    }
    return i;
}
int main()
{   char a[20]="ind";
    printf("strlen is %d \n", str-len(a));
    return 0;
}
```

String comparison.

```
int strcmp(char s1[], char s2[]);
/* compares strings s1 and s2, returns
   < 0  if s1 < s2
   = 0  if s1 == s2 (i.e. strcmp returns false)
   > 0  if s1 > s2
*/

#include <stdio.h>
#include <string.h>
int str_cmp(char s1[], char s2[])
{
    int i, iter;
    if (strlen(s1) > strlen(s2))
iter = strlen(s1)
    else
        iter = strlen(s2)
    for(i=0; i< iter ; i++)
    {
        if (s1[i] < s2[i])
            return -1;
        if (s1[i] > s2[i])
            return 1;
    }
    return 0;
}
```

```
int main()
{  char t1[] = "ind", t2[] = "ind";
   printf("cmp value = %d ", str_cmp(t1, t2));
   return 0;
}
```



Some Common Errors.

```
    printf("patter position = %d", pattern(p,t));
    return 0;
}
```

4. program to implement search and replace multiple occurances.

```
#include <stdio.h>
# include <string.h>
int pattern_match (char P[20], char T[20], int start_pos)
{   int R, S, max, K, L;
    R = strlen (P) ;  S = strlen (T);
    max = S-R;
    for (L=0 ; L<R ; L++, start_pos++)
    {
        if (P[L] != T[start_pos])
            break;
    }
    if (L==R)
        return 0;
    return -1;
}
```

```
int copy-str-match (char ostr[], char RS[], int ostr-pos)
{
        for (int i=0 ; i< strlen (RS) ; i++, ostr-pos++)
                ostr [ostr-pos] = RS [i];
        return ostr-pos ;
}
int main()
{   char T[] = "inain", PS[] = "in", RS[] = "ind",
                                ostr [20] = "_ _ _ _ _ _ _ _ ";
    int ostr-pos = 0 , T-pos = 0 , flag ;
    while (T-pos < strlen (T))
    {
        flag = pattern-match (PS, T, T-pos);
        printf ("T-pos = %d   flag = %d \n", T-pos, flag);
        if (flag == -1)
        {
            ostr [ostr-pos] = T [T-pos];
            ostr-pos ++ ;
            T-pos ++ ;
        }
        else
        {   ostr-pos = copy-str-match (ostr, RS, ostr-pos);
            T-pos = T-pos + strlen (PS);
        }
}   }
```

```
    ostr[ostr_pos] = '\0';
    printf("replaced string = %s", ostr);
    return 0;
}
```