```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <windows.h>

int xstrlen ( char * ) ;
void xstrcpy ( char *, char * ) ;
void xstrcat ( char *, char * ) ;
int xstrcmp ( char *, char * ) ;
void show ( char * ) ;

int main( )
{
    char s1[ ] = "kicit" ;
    char s2[ ] = "Nagpur" ;
    char s3[20] ;
    int len ;

    system ( "cls" ) ;
```

```
    printf ( "String s1: %s\n", s1 ) ;
    len = xstrlen ( s1 ) ;
    printf ( "length of the string s1: %d\n", len ) ;

    printf ( "String s2: %s\n", s2 ) ;

    xstrcpy ( s3, s1 ) ;
    printf ( "String s3 after copying s1 to it: %s\n", s3 ) ;

    xstrcat ( s3, s2 ) ;
    printf ( "String s3 after concatenation: %s\n", s3 ) ;

    if ( xstrcmp ( s1, s2 ) == 0 )
        printf ( "The strings s1 and s2 are similar\n" ) ;
    else
        printf ( "The strings s1 and s2 are not similar\n" ) ;

    return 0 ;
}

/* finds the length of the string */
int xstrlen ( char *s )
{
    int l = 0 ;
    while ( *s )
    {
        l++ ;
        s++ ;
    }
    return l ;
}

/* copies source string s to the target string t */
void xstrcpy ( char *t, char *s )
{
    while ( *s )
    {
```

```
            *t = *s ;
            t++ ;
            s++ ;
        }
        *t = '\0' ;
}

/* concatenates the two strings */
void xstrcat ( char *t, char *s )
{
    while ( *t )
        t++ ;
    while ( *s )
        *t++ = *s++ ;
    *t = '\0' ;
}

/* compares two strings s and t for equality */
int xstrcmp ( char *s, char *t )
{
    while ( *s == *t )
    {
        if ( ! ( *s ) )
            return 0 ;
        s++ ;
        t++ ;
    }
    return ( *s - *t ) ;
}
```

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <windows.h>

#define MAX1 6
#define MAX2 20

char masterlist[MAX1][MAX2] ;
int count ;

int add ( char *s ) ;
int find ( char *s ) ;
```

```
int main( )
{
    char yourname[MAX2] ;
    int flag ;

    system ( "cls" ) ;

    flag = add.( "akshay" ) ;
    if ( flag == 0 )
        printf ( "Unable to add string\n" ) ;

    flag = add ( "parag" ) ;
    if ( flag == 0 )
        printf ( "Unable to add string\n" ) ;

    flag = add ( "raman" ) ;
    if ( flag == 0 )
        printf ( "Unable to add string\n" ) ;

    flag = add ( "srinivas" ) ;
    if ( flag == 0 )
        printf ( "Unable to add string\n" ) ;

    flag = add ( "gopal" ) ;
    if ( flag == 0 )
        printf ( "Unable to add string\n" ) ;

    flag = add ( "rajesh" ) ;
    if ( flag == 0 )
        printf ( "Unable to add string\n" ) ;

    printf ( "Enter your name: " ) ;
    gets ( yourname ) ;
    flag = find ( yourname ) ;
    if ( flag == 1 )
        printf ( "Welcome, you can enter the palace\n" ) ;
    else
        printf ( "Sorry, you are a trespasser\n" ) ;
```

```
    return 0 ;
}

/* adds string to the array */
int add ( char *s )
{
    if ( count < MAX1 )
    {
        if ( strlen ( s ) < MAX2 )
        {
            strcpy ( &masterlist[count][0], s ) ;
            count++ ;
            return 1 ;
        }
    }

    return 0 ;
}

/* finds the given string */
int find ( char *s )
{
    int flag = 0, i ;

    for ( i = 0 ; i < count ; i++ )
    {
        if ( strcmp ( &masterlist[i][0], s ) == 0 )
        {
            flag = 1 ;
            break ;
        }
    }

    return flag ;
}
```

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <windows.h>

#define MAX 6

char *names[MAX] ;
int count ;

int add ( char * ) ;
void swap ( int, int ) ;
void show( ) ;

int main( )
{
    int flag ;

    system ( "cls" ) ;

    flag = add ( "akshay" ) ;
    if ( flag == 0 )
        printf ( "Unable to add string\n" ) ;

    flag = add ( "parag" ) ;
    if ( flag == 0 )
        printf ( "Unable to add string\n" ) ;

    flag = add ( "raman" ) ;
    if ( flag == 0 )
        printf ( "Unable to add string\n" ) ;

    flag = add ( "srinivas" ) ;
    if ( flag == 0 )
        printf ( "Unable to add string\n" ) ;

    flag = add ( "gopal" ) ;
    if ( flag == 0 )
```

```
        printf ( "Unable to add string\n" ) ;

    flag = add ( "rajesh" ) ;
    if ( flag == 0 )
        printf ( "Unable to add string\n" ) ;
    printf ( "Names before swapping:\n" ) ;
    show ( ) ;

    swap ( 2, 3 ) ;
    printf ( "Names after swapping:\n" ) ;
    show( ) ;

    return 0 ;
}

/* adds given string */
int add ( char *s )
{
    if ( count < MAX )
    {
        names[count] = s ;
        count++ ;
        return 1 ;
    }
    else
        return 0 ;
}

/* swaps the names at given two positions */
void swap ( int i, int j )
{
    char *temp ;
    temp = names[i] ;
    names[i] = names[j] ;
    names[j] = temp ;
}

/* displays the elements */
```

```
void show ( )
{
        int i ;
        for ( i = 0 ; i < count ; i++ )
                puts ( names[i] ) ;
        printf ( "\n" ) ;
}
```

Output:

```
Names before swapping:
akshay
parag
raman
srinivas
gopal
rajesh
```

malloc( ) in the array of pointers to strings. T
following program:

```
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
#include <string.h>
#include <windows.h>

int main( )
{
    char *name[5] ;
    char str[20] ;
    int i ;

    system ( "cls" ) ;

    for ( i = 0 ; i < 5 ; i++ )
    {
        printf ( "Enter a String: " ) ;
        gets ( str ) ;
        name[i] = ( char * ) malloc ( strlen ( str ) + 1 ) ;
        strcpy ( name[i], str ) ;
    }

    printf ( "\n" ) ;

    printf ( "The strings are:\n" ) ;

    for ( i = 0 ; i < 5 ; i++ )
        printf ( "%s\n", name[i] ) ;

    for ( i = 0 ; i < 5 ; i++ )
        free ( name[i] ) ;

    return 0 ;
}
```

Brute Force Algorithm

This is the simplest of all the algorithms used for pattern matc
According to this algorithm the string **s** in which the pattern s
is to be searched is scanned character by character. Beginning
the 0^{th} character of the string **s**, each character is compared
each and every character of the pattern string **p**. This pr
continues till either there is no mismatch or the pattern string
not exhausted. If a match is found then the index of the charac
s from which the comparison began is returned as the pos
where the pattern string **p** is found. However, if there is a misn
then next character of the string **s** is considered and there onv
again it is compared with the characters of the pattern strir
How long would this process continue? The process ends e
when a match is found or when the number of characters i
string **s** that remain to be scanned is less than the total numb
characters in the pattern string **p**.

Let us now see a program that implements this algorithm.

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <windows.h>

int xstrsearch ( char *, char * ) ;
void show( ) ;
int main( )
{
    char s1[ ] = "NagpurKicit" ;
    char s2[ ] = "Kicit" ;
    int pos ;
```

```
    system ( "cls" ) ;

    printf ( "String s1: %s\n\n", s1 ) ;

    printf ( "String s2: %s\n\n", s2 ) ;

    /* search if s2 is present in s1 */
    pos = xstrsearch ( s1, s2 ) ;
    printf ( "The pattern string is found at position: %d\n", pos ) ;

    return 0 ;
}

/* searches for the given pattern s2 into the string s1 */
int xstrsearch ( char * s1, char * s2 )
{
    int i, j, k ;
    int l1 = strlen ( s1 ) ;
    int l2 = strlen ( s2 ) ;

    for ( i = 0 ; i <= l1 - l2 ; i++ )
    {
        j = 0 ;
        k = i ;
        while ( ( s1[k] == s2[j] ) && ( j < l2 ) )
        {
            k++ ;
            j++ ;
        }
        if ( j == l2 )
            return i ;
    }
    return -1 ;
}
```

Output:

String s1: NagpurKicit

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <malloc.h>
#include <windows.h>

int search ( char *, char ) ;
int isequals ( char *, char * ) ;
int issmaller ( char *, char * ) ;
int isgreater ( char *, char * ) ;
char * getsub ( char *, int, int ) ;
char * leftsub ( char *, int n ) ;
char * rightsub ( char *, int n ) ;
void upper ( char * ) ;
void lower ( char * ) ;
void reverse ( char * ) ;
int replace ( char *, char, char ) ;
int setat ( char *, char, int ) ;

int main( )
{
    char s1[ ] = "Hello" ;
    char s2[ ] = "Hello World" ;
    char s3[ ] = "Four hundred thirty two" ;
    char ch, *s ;
    int i ;

    system ( "cls" ) ;

    printf ( "String s1: %s\n", s1 ) ;

    /* check for the first occurrence of a character */
    printf ( "Enter character to search: " ) ;
    scanf ( "%c", &ch ) ;
    i = search ( s1, ch ) ;
    if ( i != -1 )
```

```
        printf ( "The first occurrence of character %c is found at index no.
%d\n\n", ch, i ) ;
    else
        printf ( "Character %c is not present in the list.\n", ch ) ;

    printf ( "String s2: %s\n", s2 ) ;

    /* compares two strings s1 and s2 */
    i = isequals ( s1, s2 ) ;
    if ( i == 1 )
        printf ( "Strings s1 and s2 are identical\n" ) ;
    else
        printf ( "Strings s1 and s2 are not identical\n" ) ;
    i = issmaller ( s1, s2 ) ;
    if ( i == 1 )
        printf ( "String s1 is smaller than string s2\n" ) ;
    else
        printf ( "String s1 is not smaller than string s2\n" ) ;

    i = isgreater ( s1, s2 ) ;
    if ( i == 1 )
        printf ( "String s1 is greater than string s2\n" ) ;
    else
        printf ( "String s1 is not greater than string s2\n\n" ) ;

    /* extract characters at given position */
    printf ( "String s3: %s\n", s3 ) ;
    s = getsub ( s3, 5, 7 ) ;
    printf ( "Sub string: %s\n", s ) ;
    free ( s ) ;

    /* extract leftmost n characters */
    s = leftsub ( s3, 4 ) ;
    printf ( "Left sub string: %s\n", s ) ;
    free ( s ) ;

    /* extract rightmost n characters */
    s = rightsub ( s3, 3 ) ;
```

```
    printf ( "Right sub string: %s\n", s ) ;
    free ( s ) ;

    /*  convert string to uppercase */
    upper ( s3 ) ;
    printf ( "String in upper case: %s\n", s3 ) ;

    /* convert string to lowercase */
    lower ( s3 ) ;
    printf ( "String in lower case: %s\n", s3 ) ;

    /* reverse the given string */
    reverse ( s3 ) ;
    printf ( "Reversed string: %s\n", s3 ) ;

    /* replace first occurrence of one char with new one */
    replace ( s1, 'H' , 'M' ) ;
    printf ( "String s1: %s\n", s1 ) ;

    /* sets a char at a given position */
    i = setat ( s1, 'M', 3 ) ;
    if ( i )
        printf ( "String s1: %s\n", s1 ) ;
    else
        printf ( "Invalid position.\n" ) ;

    return 0 ;
}

/* check for the first occurrence of a character */
int search ( char *str, char ch )
{
    int i = 0 ;

    while ( *str )
    {
        if ( *str == ch )
            return i ;
```

```
            str++ ;
            i++ ;
        }
        return -1 ;
}

/* checks whether two strings are equal */
int isequals ( char *s, char *t )
{
    while ( *s || *t )
    {
        if ( *s != *t )
            return 0 ;
        s++ ;
        t++ ;
    }
    return 1 ;
}

/* checks whether first string is less than second */
int issmaller ( char *s, char *t )
{
    while ( *t )
    {
        if ( *s != *t )
        {
            if ( *s < *t )
                return 1 ;
            else
                return 0 ;
        }
        t++ ;
        s++ ;
    }
    return 0 ;
}

/* checks whether first string is greater than second */
```

```
int isgreater ( char *s, char *t )
{
    while ( *s )
    {
        if ( *s != *t )
        {
            if ( *s > *t )
                return 1 ;
            else
                return 0 ;
        }
        s++ ;
        t++ ;
    }
    return 0 ;
}

/* extracts the character at given position */
char * getsub ( char *str, int spos, int n )
{
    char *s = str + spos ;
    char *t = ( char * ) malloc ( n + 1 ) ;
    int i = 0 ;

    while ( i < n )
    {
        t[i] = *s ;
        s++ ;
        i++ ;
    }
    t[i] = '\0' ;

    return t ;
}

/* extracts leftmost n characters from the string */
char * leftsub ( char *s, int n )
{
```

```
    char *t = ( char * ) malloc ( n + 1 ) ;
    int i = 0 ;

    while ( i < n )
    {
        t[i] = *s ;
        s++ ;
        i++ ;
    }
    t[i] = '\0' ;

    return t ;
}

/* extracts rightmost n characters from the string */
char * rightsub ( char *str, int n )
{
    char *t = ( char * ) malloc ( n + 1 ) ;
    int l = strlen ( str ) ;
    char *s = str + ( l - n ) ;
    int i = 0 ;

    while ( i < n )
    {
        t[i] = *s ;
        s++ ;
        i++ ;
    }
    t[i] = '\0' ;

    return t ;
}

/* converts string to uppercase */
void upper ( char *s )
{
    while ( *s )
    {
```

```
            if ( *s >= 97 && *s <= 123 )
                *s -= 32 ;
            s++ ;
        }
}

/* converts string to lowercase */
void lower ( char *s )
{
    while ( *s )
    {
        if ( *s >= 65 && *s <= 91 )
            *s += 32 ;
        s++ ;
    }
}

/* reverses a string */
void reverse ( char *str )
{
    int l = strlen ( str ) ;
    char ch, *t = ( str + l - 1 ) ;
    int i = 0 ;

    while ( i < l / 2 )
    {
        ch = *str ;
        *str = *t ;
        *t = ch ;

        str++ ;
        t-- ;
        i++ ;
    }
}

/* replaces the first occurrence of char with new char */
int replace ( char *str, char oldch, char newch )
```

```
{
    while ( *str )
    {
        if ( *str == oldch )
        {
            *str = newch ;
            return 1 ;
        }
        str++ ;
    }
    return 0 ;
}

/* sets a char at a given position */
int setat ( char *str, char ch, int i )
{
    int len = strlen ( str ) ;
    if ( i < 0 || len < i )
        return 0 ;
    *( str + i ) = ch ;
    return 1 ;
}
```