

## 1. String functions

### Aim:

To write a program in C that implements the following as user-defined functions.

- strcat(str1, str2) Appends str2 to str1
- strncpy(dest, src, n) Copies up to n characters from src to dest string
- strchr(str1, ch) Scans the string str1 for the first occurrence of the character ch and returns the position
- strset(str1, ch) Sets all characters in the string str1 to the character ch
- strcmpi(str1, str2) Compares str1 and str2 ignoring the case sensitivity and returns the difference

### Code:

```
/* Implement the following as user-defined functions.
a.strcat(str1, str2) Appends str2 to str1
b.strncpy(dest, src, n) Copies up to n characters from src to dest string
c.strchr(str1, ch) Scans the string str1 for the first occurrence of the character
ch and returns the position
d.strset(str1, ch) Sets all characters in the string str1 to the character ch
e.strcmpi(str1, str2) Compares str1 and str2 ignoring the case sensitivity and
return */
#include <stdio.h>
#include <stdlib.h>

#define wlim 20

int strl(char str[]){
    int i;
    for (i=0; str[i]!='\0'; i++);
    return i;
}

char *strncpy(char dest[], char src[], int n){
    for (int i=0; i<n; i++){
        dest[i]=src[i];
    }
    //returning the destination string
    return dest;
}

char *strchar(char str1[], char ch){
    for (char *i=str1; i<str1+strl(str1); i++){
        if (*i==ch)
            return i; //returning the pointer to the character
    }
    printf("Character not found!");
    return NULL; //returning null pointer if ch not found
}
```

```
char *strjoin(char str1[], char str2[]){
    //to find the length of each string
    int l1=strl(str1), l2=strl(str2);

    //to concatenate the strings (add str2 to str1)
    char str3[l1+l2];
    for (int i=0; i<l1; i++){
        str3[i]=str1[i];
    }
    for (int i=0; i<l2; i++){
        str3[l1+i]=str2[i];
    }
    strncpy(str1,str3,l1+l2);

    //returning the concatenated string
    return str1;
}

char *strsetch(char str1[], char ch){
    for (int i=0; str1[i]!='\0'; i++){
        str1[i]=ch;
    }
    //returning the modified string
    return str1;
}

char *strlower(char str[]){
    for (int i=0; str[i]!='\0'; i++){
        if (str[i]>=65 && str[i]<=90)
            str[i]=(char)((int)str[i]+32);
    }
    return str;
}

int strcmpi(char str1[], char str2[]){
    int diff, i;
    //to convert both strings to lower case
    //assigning them to local strings within the function so as to not change the original
    strings
    char *s1=strlower(str1);
    char *s2=strlower(str2);
    //checking the difference of the two strings
    for (i=0; s1[i]!='\0'; i++){
        diff=(int)s1[i]-(int)s2[i];
        if (diff){
            return diff;
        }
    }
}
```

```
//if second string is larger than the first
if (s2[i]!='\0')
    diff--(int)s2[i];
//here, returns diff=0 if both strings are equal
return diff;
}

int main() {
    char s[wlim], b[wlim]={};
    printf("Enter string: ");
    scanf("%s",s);

    int ch;
    printf("Enter choice: ");
    scanf("%d",&ch);

    switch(ch){
        case 1: {
            printf("Implementing strcat():-\n");
            char t[wlim]={};
            printf("Enter second string: ");
            scanf("%s",t);
            printf("Appending str2 to str1: %s\n",strjoin(s,t));
            break;
        }

        case 2: {
            printf("Implementing strncpy():-\n");
            char t[wlim]={};
            printf("Enter second string: ");
            scanf("%s",t);
            int n;
            printf("Enter no. of characters to copy: ");
            scanf("%d",&n);
            printf("Copying first n characters of source to destination string:
%s\n",strncpy(s,t,n));
            break;
        }

        case 3: {
            printf("Implementing strchr():-\n");
            char ch;
            printf("Enter required character: ");
            scanf(" %c",&ch);
            printf("Returning the pointer to the first occurrence of the given character:
%s\n",strchr(s,ch));
            break;
        }
    }
}
```

```
case 4: {
    printf("Implementing strset():-\n");
    char ch;
    printf("Enter required character: ");
    scanf(" %c",&ch);
    printf("Setting all the characters of the string to the given character:
%s\n",strsetch(s,ch));
    break;
}

case 5: {
    printf("Implementing strcmpi():-\n");
    char t[wlim]={};
    printf("Enter second string: ");
    scanf("%s",t);
    printf("Comparing the two strings while ignoring case...\nReturn value:
%d\t\t",strcmpi(s,t));
    if (strcmpi(s,t)==0) printf("Hence, equivalent.\n");
    else printf("Hence, not equivalent.\n");
    break;
}
default: {
    printf("Invalid choice! Exiting...");
    return 0;
}
}

return 0;
}
```

**Output:**

```
> gcc -o q1.o q1.c
> ./q1.o
Enter string: con
Enter choice: 1
Implementing strcat():-
Enter second string: test
Appending str2 to str1: contest
> ./q1.o
Enter string: name
Enter choice: 2
Implementing strncpy():-
Enter second string: cone
Enter no. of characters to copy: 2
Copying first n characters of source to destination string: come
> ./q1.o
Enter string: interest
Enter choice: 3
Implementing strchr():-
Enter required character: e
Returning the pointer to the first occurrence of the given character: erest
> ./q1.o
Enter string: password
Enter choice: 4
Implementing strset():-
Enter required character: *
Setting all the characters of the string to the given character: *****
```

```
> ./q1.o
Enter string: Hello
Enter choice: 5
Implementing strcasecmp():-
Enter second string: hello
Comparing the two strings while ignoring case...
Return value: 0 Hence, equivalent.
> ./q1.o
Enter string: Hello
Enter choice: 5
Implementing strcasecmp():-
Enter second string: world
Comparing the two strings while ignoring case...
Return value: -15 Hence, not equivalent.
```

**Result:**

A program for implementing some standard C string functions as user-defined functions is written and executed.

## 2. First Occurrence of Substring

### Aim:

To write a program that searches for the first occurrence of a substring in a given string and returns the position of the first character of the first occurrence.

### Code:

```
//to search for the first occurrence of a substring in a given string
#include <stdio.h>

#define wlim 25

int search(char s[], char subs[]){
    int i, flag;
    for (i=0; s[i]!='\0'; i++){
        if (s[i]==subs[0]){
            int k=i; flag=0;
            for (int j=0; subs[j]!='\0'; j++){
                if (s[k++]!=subs[j]){
                    flag=1; break;
                }
            }
            if (flag==0)
                return i;
        }
    }
    return -1;
}

int main(){
    char str[wlim], sub_str[wlim];
    printf("Enter string: ");
    scanf("%s",str);

    printf("Enter substring: ");
    scanf("%s",sub_str);

    if (search(str,sub_str)==-1)
        printf("substring not found\n");
    else
        printf("First occurrence of given substring in the string: position
%d\n",search(str,sub_str)+1);

    return 0;
}
```

**Output:**

```
> gcc -o q2.o q2.c
> ./q2.o
Enter string: wonderful
Enter substring: ful
First occurrence of given substring in the string: position 7
> ./q2.o
Enter string: wonderful
Enter substring: ponder
substring not found
>
```

**Result:**

A program for finding the first occurrence of a substring in a given string is written and executed.

### 3. Reversing a String

**Aim:**

To write a program to reverse a given string without using library functions with minimum number of exchanges done while the source string is being modified.

**Code:**

```
//to reverse a given string
//no extra string to be used, source string to be modified
//no. of exchanges should be minimum
#include <stdio.h>

#define wlim 15

//function to calculate the string length
int strl(char str[]){
    int i;
    for (i=0; str[i]!='\0'; i++);
    return i;
}

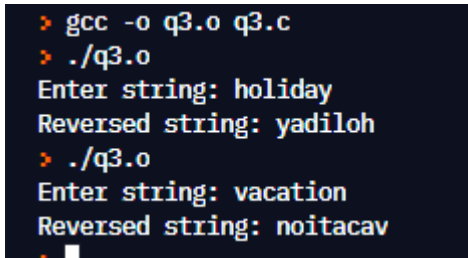
int main(){
    char str[wlim];
    printf("Enter string: ");
    scanf("%s",str);

    //to reverse the string
    char temp;
    int l=strl(str);
    for (int i=0; str[i]!='\0'; i++){
        //splitting for l = odd or even to perform minimum exchanges
        if (l%2==0){
            if (i<l/2){
                temp=str[i];
                str[i]=str[l-1-i];
                str[l-1-i]=temp;
            }
        }
        else{
            if (i<(l-1)/2){
                temp=str[i];
                str[i]=str[l-1-i];
                str[l-1-i]=temp;
            }
        }
    }
}
```



```
//to print the reversed string
printf("Reversed string: %s\n",str);

return 0;
}
```

**Output:**

```
> gcc -o q3.o q3.c
> ./q3.o
Enter string: holiday
Reversed string: yadiloh
> ./q3.o
Enter string: vacation
Reversed string: noitacav
```

**Result:**

A program for reversing a given string with minimal number of exchanges is written and executed.

## 4. Encoding and Decoding

### Aim:

To write a program that encodes and decodes a given line of text as per the given instructions.

### Code:

```
//to encode or decode a line of text
/* To encode a line of
text, proceed as follows.
a) Convert each character, including blank spaces, to its ASCII equivalent.
b) Generate a positive random integer. Add this integer to the ASCII equivalent of each
character. The same random integer will be used for the entire line of text. After adding
make
sure the integer falls within ASCII range, so that the encoded character is always an
ASCII
character
c) Display the characters that correspond to the encoded ASCII values.
The procedure is reversed when decoding the line of text and also print the decoded
text.*/
```

```
#include <stdio.h>
#include <stdlib.h>
```

```
#define tlim 50
int randnum;
```

```
void encode(char s[]){
    randnum=rand() % 128;
    for (int i=0; s[i]!='\0'; i++){
        s[i]=(int)s[i];
        s[i]+=randnum;
        if (s[i]>127)
            s[i]-=128;
        s[i]=(char)s[i];
    }
}
```

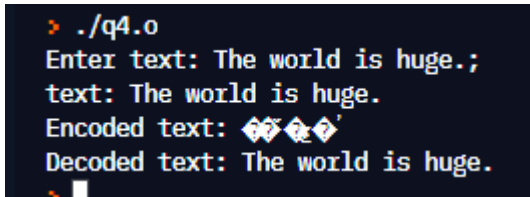
```
void decode(char s[]){
    for (int i=0; s[i]!='\0'; i++){
        s[i]=(int)s[i]-randnum;
        if (s[i]<0)
            s[i]+=128;
        s[i]=(char)s[i];
    }
}
```

```
int main(){
    char text[tlim];
    printf("Enter text: ");
    scanf("%[^\n]s",text);
    printf("text: %s\n",text);

    encode(text);
    printf("Encoded text: %s\n",text);

    decode(text);
    printf("Decoded text: %s\n",text);

    return 0;
}
```

**Output:**

```
> ./q4.o
Enter text: The world is huge.;
text: The world is huge.
Encoded text: 
Decoded text: The world is huge.
```

**Result:**

A program to encode and decode a given line of text is written and executed.