LABORATORY WORKS FOR TOC

LAB 1 : Write a program to find prefixes, suffixes and substring from given string.

Code:

```
/* To find substring, prefix, suffix of a string */
#include<stdio.h>
#include<string.h>
void find_prefix(char string[]);
void find_suffix(char string[]);
void find_substring(char string[],int,int);

int main()
{
        char string[20];
        int i,j;
        printf("\n Enter a string\t");
        gets(string);

        printf("\n Prefixes:");
        find_prefix(string);
        printf("\n Suffixes");
        find_suffix(string);

        printf("\nEnter i and j for substring");
        scanf("%d%d",&i,&j);
        find_substring(string,i,j);

        return 0;
}

void find_prefix(char string[])
{
        int i,j;
        char prefix[20];
        for(i=strlen(string);i>=0;i--)
        {
                for( j = 0; j<i;j++)
                 {
```

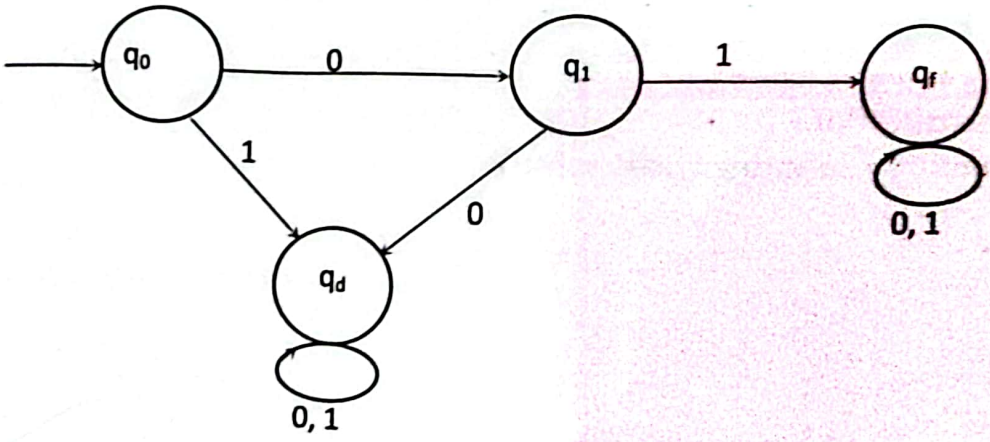
```
                prefix[j]= string[j];
            }
            prefix[j]='\0';
            printf("\n %s",prefix);
        }
}
void find_suffix(char string[])
{
        int i,j,k;
        char suffix[20];
        for(i=0;i<=strlen(string);i++)
        {
            k = i;
            for( j = 0; j<strlen(string);j++)
             {
                suffix[j]= string[k];
                k++;
             }
             suffix[j]='\0';
             printf("\n %s",suffix);
        }
}

void find_substring(char string[],int x, int y)
{
        char substr[20];

        int k=0;
        for(int i=x-1;i<y;i++)
        {
           substr[k]=string[i];
           k++;
        }
        substr[k]='\0';
        printf("\n Substring:\n%s",substr);

}
```

OUTPUT:

```
C:\Users\c\Desktop\lab1sttring.exe
computat
computa
comput
compu
comp
com
co
c

Suffixes
computation
omputation
mputation
putation
utation
tation
ation
tion
ion
on
n

Enter i and j for substring
2  5

 Substring:
ompu
--------------------------------
Process exited after 61.09 seconds with return value 0
Press any key to continue . . .
```

LAB 2: Write program to implement following DFA's over alphabet $\Sigma = \{0, 1\}$.

The DFA that accepts all the strings that start with 01.



ode

```
Implement a DFA for L = { set of all strings over {0,1} such that string start with 01 */
include<stdio.h>
num states { q0, q1, qf,qd};
num states delta(enum states, char);

t main()

        char input[20];
        enum states curr_state = q0;
        int i =0;

        printf("\n Enter a binary string\t");
        gets(input);
        char ch = input[i];
        while( ch !='\0')
        {
           curr_state = delta(curr_state,ch);
                 ch = input[++i];
        }

        if(curr_state == qf)
          printf("\n The string %s is accepted.",input);
        else
           printf("\n The string %s is not accepted.",input);
  return 0;
}
// Transition Function
enum states delta(enum states s, char ch)
{
     enum states curr_state;
           switch(s)
                 {
                       case q0:
                             if(ch=='0')
                               curr_state = q1;
                             else
                               curr_state = qd;
                             break;
                       case q1:
                             if(ch=='1')
                               curr_state = qf;
                             else
```

```
                    curr_state = qd;
                    break;
            case qf:
                    if(ch=='0')
                      curr_state = qf;
                    else
                      curr_state = qf;
                    break;
            case qd:
                    if(ch=='0')
                      curr_state = qd;
                    else
                      curr_state = qd;
                    break;
          }
    return curr_state;
}
```

OUTPUT:

```
Enter a binary string  01110

The string 01110 is accepted.
--------------------------------
Process exited after 6.659 seconds with return value 0
Press any key to continue . . .
```

```
Enter a binary string  1101

The string 1101 is not accepted.
--------------------------------
Process exited after 5.392 seconds with return value 0
Press any key to continue . . .
```

AB 3: The DFA that accepts all the strings that end with 01.

Q = { q0, q1, qf}
start state = q0,
Final state = qf
Transition function, δ is defined as:

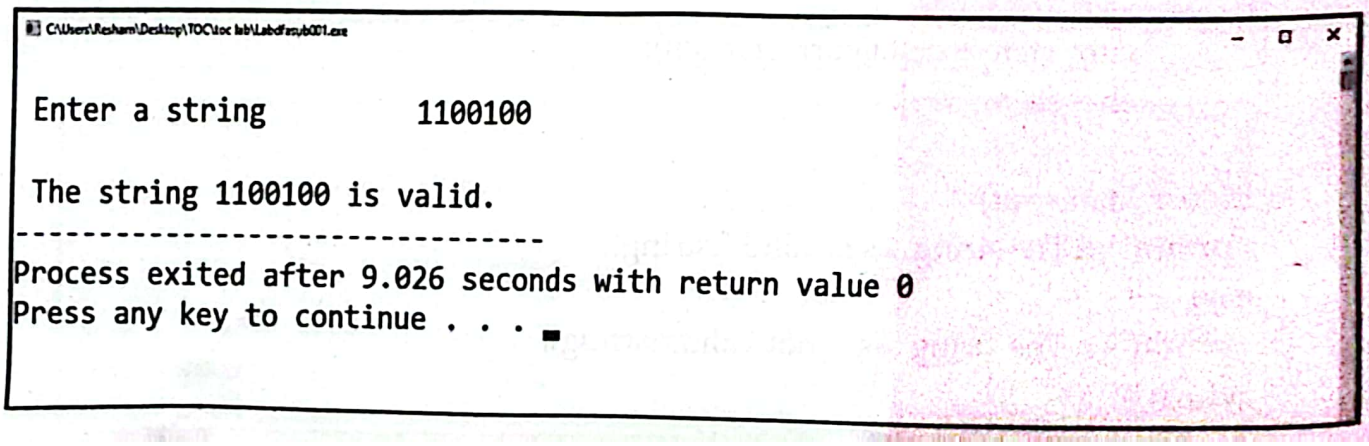
$\delta\ (q0, 0) = q1$
$\delta\ (q0, 1) = q0$
$\delta\ (q1, 0) = q1$
$\delta\ (q1, 1) = qf$
$\delta\ (qf, 0) = q1$
$\delta\ (qf, 1) = q0$

Code:

```
// Implement a DFA for L = { set of all strings over {0,1} such that string end with 01
#include<stdio.h>

enum states { q0, q1, qf};
enum states delta(enum states, char);

int main()
{
        char input[20];
        enum states curr_state = q0;
        int i =0;

        printf("\n Enter a binary string\t");
        gets(input);
        char ch = input[i];
        while( ch !='\0')
        {
           curr_state = delta(curr_state,ch);
                ch = input[++i];
        }

        if(curr_state == qf)
          printf("\n The string %s is accepted.",input);
        else
           printf("\n The string %s is not accepted.",input);

  return 0;
}
```

```
enum states delta(enum states s, char ch)
{
        enum states curr_state;
        switch(s)
        {
                case q0:
                        if(ch=='0')
                          curr_state = q1;
                        else
                          curr_state = q0;
                        break;
                case q1:
                        if(ch=='0')
                          curr_state = q2;
                        else
                          curr_state = q0;
                        break;

                case q2:
                        if(ch=='0')
                          curr_state = q2;
                        else
                          curr_state = qf;
                        break;
                case qf:
                        if(ch=='0'||ch=='1')
                          curr_state = qf;
                }
                return curr_state;
}
```

OUTPUT



```
Enter a string        11010

The string 11010 is not valid.
--------------------------------
Process exited after 6.237 seconds with return value 0
Press any key to continue . . .
```

LAB 5: Write a program to validate C identifiers and keywords.

C identifiers: These are the names of variables, functions, arrays, structures and pointers etc. The first character of C identifiers must be letter or underscore and remaining characters might be letters, digits or underscore.

Keywords: These are the reserved words having predefined meaning in the language. There are 32 keywords in C. They cannot be used as identifiers.

code:

```
// to identify valid identifiers and keywords in C
#include<stdio.h>
#include<string.h>
char keyword[32][10] = {"auto","double","int","struct","break","else","long",
switch","case", "enum", "register", "typedef","char", "extern", "return", "union", "const",
"float", "short", "unsigned","continue","for","signed","void","default", "goto", "sizeof",
"volatile", "do","if","static","while"} ;
enum states { q0, qf, qd};
enum states delta(enum states, char);
int iskeyword(char []);
int main()
{
        enum states curr_state = q0;
        char string[20], ch;
        int i=0;

        printf("\n Enter a string \t");
        gets(string);

        ch = string[i];
```

```
        if(iskeyword(string))
          printf("\n The string %s is keyword.",string);
        else
        {
                while(ch!='\0')
                {
                        curr_state = delta(curr_state,ch);
                        ch = string[++i];
                }
                if(curr_state==qf)
                        printf("\n The string %s is valid indentifier.",string);
                else
                        printf("\n The string %s is neither keyword nor valid
identifier.",string);
        }
        return 0;
} //end of the main

//transition function
enum states delta(enum states s, char ch)
{
        enum states curr_state;
        switch(s)
        {
                case q0:
                        if(ch>='A' && ch<='Z'||ch>='a'&&ch<='z'||ch=='_')
                          curr_state = qf;
                        else
                          curr_state = qd;
                        break;
                case qf:
                        if(ch>='A' &&
ch<='Z'||ch>='a'&&ch<='z'||ch=='_'||ch>='0'&&ch<='9')
                          curr_state = qf;
                        else
                          curr_state = qd;
                        break;
                case qd:
                        curr_state = qd;
        }
```

```
        return curr_state;
}

int iskeyword(char str[])
{
        for(int i=0;i<32;i++)
        {
                if(strcmp(str,keyword[i])==0)
                  return 1;
        }
        return 0;

}
```

OUTPUT

```
Enter a string          num

The string num is valid indentifier.
--------------------------------
Process exited after 10.43 seconds with return value 0
Press any key to continue . . .
```

```
Enter a string          1abc

The string 1abc is neither keyword nor valid identifier.
--------------------------------
Process exited after 8.069 seconds with return value 0
Press any key to continue . . .
```

```
Enter a string          int

The string int is keyword.
--------------------------------
Process exited after 2.508 seconds with return value 0
Press any key to continue . . .
```