```c
//takes a string as input and verifies if it is a valid keyword or identifier.

// Note: valid keywords are : enum, int, char, float, struct, union, if, else, while, for, switch, goto, short,
long.

// an identifier starts with letter or underscore

// after first character any sequence of letters, digits and underscore can occur.


#include <stdio.h>

#include <string.h>


int isidentifier(char input[]);


void main() {

  char kwlist[14][10] = {"enum", "int", "char", "float", "struct", "union", "if", "else", "while", "for",
"switch", "goto", "short", "long"};

  int i, res;

  char input[10];


  printf("Enter a string: ");

  scanf("%s", input);


  for (i = 0; i < 14; i++) {

    if (strcmp(kwlist[i], input) == 0) {

      printf("\nA keyword\n");

      return;

    }

  }


  res = isidentifier(input);

  if (res == 1) {
        printf("\nIdentifier\n");
    } else {
        printf("\nInvalid identifier\n");
    }
}

int isidentifier(char input[]) {
    int i = 0, curr_state = 0;
    char c;

    while (input[i] != '\0') {
        c = input[i];
        switch (curr_state) {
            case 0:
                if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c == '_'))
                    curr_state = 1;
                else
                    curr_state = 2;
                break;
            case 1:
                if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c == '_') || (c >= '0' && c <= '9'))
                    curr_state = 1;
                else
                    curr_state = 2;
                break;
        }
        i++;
    }
    if (curr_state == 2)
        return 0;
    else
        return 1;
}

//dfa that accepts strings of 0 and 1 that are greater than 2 in length
#include<stdio.h>
#include<string.h>

void main()
{
    char w[20];
    int i, len;
    char cur_state = '0', c;

    printf("\nEnter a string:\n");
    scanf("%s", w);  // Use scanf instead of gets
    len = strlen(w);

    for(i = 0; i < len; i++)
    {
        c = w[i];
        switch(cur_state)
        {
            case '0': cur_state = '1'; break;
            case '1': cur_state = '2'; break;
            case '2': cur_state = '2'; break;
            default: printf("\nInvalid state!!!");
            return;
        }
    }

    if(cur_state == '2')
    {
        printf("\nString accepted!!!\n");
    }
    else
    {
        printf("\nString rejected!!!\n");
    }
}

/*wap in c that implements a nfa for L={w belongs to {0,1)* | w ends with 01}*/

#include<stdio.h>
#include<string.h>

char input[20];
int l, flag;

int q0(int i)
{
              int k= i;
              if(i<l)
              {
                            if (input[i]=='0')
                            {
```

```
                        k++;
                        q0(k); q1(k);
                }
                else
                {
                        if (input[i]=='1')
                        {
                                i++; q0(i);
                        }
                }
        }
}

int q1(int i)
{
        if(i<l)
        {
                if(input[i]=='1')
                {
                        i++; q2(i);
                }
        }
}

int q2(int i)
{
        if(input[i]=='\0')
        {
                flag=1;
        }
}

void main()
{
        printf("\n enter a string:\n");
        scanf("%s",input);
        l=strlen(input);

        int i=0;
        flag=0;
        q0(i);

        if(flag==1)
        {
                printf("\n accepted\n");
        }
        else
        {
                printf("\nrejected\n");
        }
}
```

```
// c implementation of push down automata(pda) that accepts language L=L = {w | w belongs to {0,1}*,
|w0| = |w1|}.

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

```
void maketransition(char, char, int);

void push(char);

void pop();

char gettop();


int current = 0;

char STACK[20];

int top = -1;


void push(char c) {

   if (top == 19) {

      printf("Stack full!!!\n");

      return;

   }

   STACK[++top] = c;

}


char gettop() {

   if (top == -1) {

      return '$';

   }

   return STACK[top];

}


void pop() {

   if (top == -1) {

      printf("Stack empty\n");

      return;

   }
```

```
    top--;
}

void maketransition(char c, char st, int state) {
    switch (state) {
        case 0:
            if (c == 'e' && st == 'e') {
                push('$');
                current = 1;
            }
            break;

        case 1:
            if (c == '0' && (st == '$' || st == '0')) {
                push('0');
                current = 1;
            } else if (c == '1' && st == '0') {
                pop();
                current = 1;
            } else if (c == '1' && (st == '$' || st == '1')) {
                push('1');
                current = 1;
            } else if (c == '0' && st == '1') {
                pop();
                current = 1;
            } else if (c == 'e' && st == '$') {
                pop();
                current = 2;
            }
            break;

        case 2:
            break;
    }
}

int main() {
    char inputstr[20], c;
    int i = 0;

    printf("Enter a string: ");
    scanf("%s", inputstr);

    maketransition('e', 'e', current);
    c = inputstr[i];

    while (c != '\0') {
        maketransition(c, gettop(), current);
        c = inputstr[++i];
    }

    maketransition('e', gettop(), current);

    if (current == 2 && top == -1)
              {
        printf("\nAccepted!!!\n");
    }
              else
            {
        printf("\nRejected!!!\n");
    }

    return 0;
}

//to take a string and 2 indices i and j as input and print the substring between i and j

#include <stdio.h>
#include <string.h>

int main() {
    char str[100];
    int i, j, k;

    printf("Enter a string: ");
    scanf("%s", str);

    printf("Enter the starting index i: ");
    scanf("%d", &i);
    printf("Enter the ending index j: ");
    scanf("%d", &j);

    int length = strlen(str);
    if (i < 0 || j >= length || i > j) {
        printf("Invalid indices.\n");
        return 1;
    }
    printf("Substring between index %d and %d: ", i, j);
    for (k = i; k <= j; k++) {
        printf("%c", str[k]);
    }
    printf("\n");

    return 0;
}

//wap to take string as input and print its prefixes and suffixes
#include<stdio.h>
#include<conio.h>
#include<string.h>

void printsuffix(char[]);
void printprefix(char[]);

void main()
{
    char str[20];
    printf("Enter a word: ");
    scanf("%s", str); // Replacing gets with scanf to read a word
    printf("\nSuffixes:\n.....\n");
    printsuffix(str);
    printf("\n------------------------\n");
    printf("\nPrefixes:\n.....\n");
    printprefix(str);
}
```

```
void printsuffix(char w[])
{
    int len= strlen(w);
    int i, j;
    for(i=0; i<len; i++)
    {
        for(j=i; j<len; j++)
        {
            printf("%c",w[j]);
        }
        printf("\n");
    }
}

void printprefix(char w[])
{
    int len= strlen(w);
    int i, j;
    for(i= len; i>=0; i--)
    {
        for(j=0; j<i; j++)
        {
            printf("%c",w[j]);
        }
        printf("\n");
    }
}
```

```
//turing machine

#include<stdio.h>

#include<string.h>


int current=0;


void main()

{

    char inputstr[20], c;

    int i=0;


    // Properly initialize the input string

    for(i=0; i<20; i++)

    {

        inputstr[i]='\0';

    }


    printf("\nEnter an input string:\n");

    scanf("%s",inputstr);


    i = 0; // Reset index to start at the beginning of the input string


    while(1)

    {

        c = inputstr[i];

        switch(current)
```

```
{
    case 0:
        if(c == '0')
        {
            inputstr[i] = 'X';
            i++;
            current = 1;
        }
        else if(c == 'Y')
        {
            i++;
            current = 3;
        }
        else
        {
            current = -1;
        }
        break;

    case 1:
        if(c == '0')
        {
            i++;
            current = 1;
        }
        else if(c == 'Y')
        {
            i++;
            current = 1;
    }
    else if(c == '1')
    {
        inputstr[i] = 'Y';
        i--;
        current = 2;
    }
    else
    {
        current = -1;  // Reject if invalid input found
    }
    break;

case 2:
    if(c == '0' || c == 'Y')
    {
        i--;
        current = 2;
    }
    else if(c == 'X')
    {
        i++;
        current = 0;
    }
    else
    {
        current = -1;  // Reject on invalid input
    }
    break;
```

```
        case 3:
          if(c == 'Y')
          {
            i++;
            current = 3;
          }
          else if(c == '\0')  // End of string reached
          {
            current = 4;
          }
          else
          {
            current = -1;  // Reject on invalid input
          }
          break;
      }
      if (current == -1 || current == 4)
      {
        break;
      }
    }

    if (current == 4)
    {
      printf("\nString Accepted!!!!!\n");
    }
    else
    {
        printf("\nString Rejected!!!!!\n");
    }
}
```