Lab Assignment

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

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

int main(){
char string[999],temp[999]="",pre[999]="",suff[999]="",substr[999]="";
int i,j;
char state='A';//States : A,B,C,D
printf("Enter the string :: ");
scanf("%s",string);

printf("\n-->PREFIX :\n");
for(i=0;i<strlen(string);i++){
strncat(pre,&string[i],1);
printf("%s\n",pre);
}
printf("\n-->SUFFIX :\n");
for(i=strlen(string)-1;i>=0;i--){
strncat(suff,&string[i],1);
strcpy(temp,suff);
printf("%s\n",strrev(temp));
}
printf("\n-->SUBSTRING :\n");
for(i=0;i<strlen(string);i++){
for(j=i;j<strlen(string);j++){
strncat(substr,&string[j],1);
printf("%s\t",substr);
}
printf("\n");
strcpy(substr,"");
}

return 0;
}
```

2. Write a program to implement DFA such that it accepts the string that starts with 01, over alphabet Σ={0,1}.

```
#include <stdio.h>

int main(){
char input[999];
int i;
char state='A';//States : A,B,C,D
printf("Enter the string to be checked [INPUTS:0,1] :: ");
scanf("%s",input);
```

```
for(i=0;input[i]!='\0';i++){
switch(state){

case 'A':
if(input[i]=='0')
state='B';
else if(input[i]=='1')
state='D';
break;

case 'B':
if(input[i]=='1')
state='C';
else if(input[i]=='0')
state='D';
break;

case 'C':
if(input[i]=='0')
state='C';
else if(input[i]=='1')
state='C'; // ?
break;

case 'D':
if(input[i]=='0')
state='D';
else if(input[i]=='1')
state='D';
break;



}
}
printf("\n");
if(state=='C')
printf("-->The string is accepted over DFA.\n");
else
printf("-->The string is not accepted over DFA.\n");
return 0;
}
```

3. Write a program to implement DFA such that it accepts the string that ends with 01, over alphabet Σ={0,1}.

```
#include <stdio.h>

int main(){
char input[999];
int i;
```

```
char state='A';//States : A,B,C,D
printf("Enter the string to be checked [INPUTS:0,1] :: ");
scanf("%s",input);
for(i=0;input[i]!='\0';i++){
switch(state){

case 'A':
if(input[i]=='0')
state='B';
else if(input[i]=='1')
state='A';
break;

case 'B':
if(input[i]=='0')
state='B';
else if(input[i]=='1')
state='C';
break;

case 'C':
if(input[i]=='0')
state='B';
else if(input[i]=='1')
state='A';
break;
}
}
printf("\n");
if(state=='C')
printf("-->The string is accepted over DFA.\n");
else
printf("-->The string is not accepted over DFA.\n");
return 0;
}
```

4. Write a program to implement DFA such that it accepts the string that contains substring 001, over alphabet Σ={0,1}.

```
#include <stdio.h>

int main(){
char input[999];
int i;
char state='A';//States : A,B,C,D
printf("Enter the string to be checked [INPUTS:0,1] :: ");
scanf("%s",input);
for(i=0;input[i]!='\0';i++){
switch(state){
```

```
case 'A':
if(input[i]=='0')
state='B';
else if(input[i]=='1')
state='A';
break;

case 'B':
if(input[i]=='1')
state='A';
else if(input[i]=='0')
state='C';
break;

case 'C':
if(input[i]=='0')
state='C';
else if(input[i]=='1')
state='D';
break;

case 'D':
if(input[i]=='0')
state='D';
else if(input[i]=='1')
state='D';
break;



}
}
printf("\n");
if(state=='D')
printf("-->The string is accepted over DFA.\n");
else
printf("-->The string is not accepted over DFA.\n");
return 0;
}
```

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

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

int main(){
char keyword_list[32][10]={ "auto","break","case","char",
"const","continue","default","do",
"double","else","enum","extern",
"float","for","goto","if",
```

```
"int","long","register","return",
"short","signed","sizeof","static",
"struct","switch","typedef","union",
"unsigned","void","volatile","while"
};
char input[10];
int i, result=0;
printf("Enter the string to be checked :: ");
scanf("%s",input);
for(i=0;i<32;i++){
if(strcmp(input,keyword_list[i])==0){
printf("\n->%s is keyword\n",input);
goto end;
}
}
char c = input[0];
if((c=='_')||(c>='A' && c<='Z')||(c>='a' && c<='z')){
for(i=1;input[i]!='\0';i++){
c=input[i];
if((c=='_')||(c>='A' && c<='Z')||(c>='a' && c<='z')||(c<='0'||c<='9'))
result = 1;
else
result = 0;
}
}
if(result==1){
printf("\n->%s is identifier\n",input);
}
else{
printf("\n-> Invalid identifier\n");
}

end:
return 0;
}
```

6. Write a program to implement NFA such that it accepts the string that starts with 01, over alphabet Σ={0,1}.

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

int main(){
char input[999];
int i;
char state='A';//States : A,B,C
printf("Enter the string to be checked [INPUTS:0,1] :: ");
scanf("%s",input);
for(i=0;input[i]!='\0';i++){
switch(state){

case 'A':
```

```
if(input[i]=='0')
state='B';
else if(input[i]=='1')
goto end;
break;


case 'B':
if(input[i]=='1')
state='C';
else if(input[i]=='0')
goto end;
break;


case 'C':
if(input[i]=='0')
state='C';
else if(input[i]=='1')
state='C';
break;


}
}
printf("\n");
if(state=='C'){
printf("-->The string is accepted over NFA.\n");
}
else{
end:
printf("\n-->The string is not accepted over NFA.\n");
}


return 0;
}
```

7. Write a program to implement NFA such that it accepts the string that ends with 01, over alphabet Σ={0,1}.

```
#include <stdio.h>


char input[999];
int NFA(int, char);
int main(){
char state;
printf("Enter the string to be checked [INPUTS:0,1] :: ");
scanf("%s",input);


state=NFA(0, 'A');


printf("\n");
```

```
if(state=='C')
printf("-->The string is accepted over NFA.\n");
else
printf("-->The string is not accepted over NFA.\n");
// getch();
return 0;
}
int NFA(int index,char state){
if(input[index]=='\0'){
return state;
}
if(state=='A'){
if(input[index]=='0'){
char next_state1=NFA(index+1,'A');
char next_state2=NFA(index+1,'B');
if((next_state1=='C')||(next_state2=='C'))
return 'C';
}
else if(input[index]=='1')
return NFA(index+1,'A');
}
else if(state=='B'){
if(input[index]=='1')
return NFA(index+1,'C');
}
return 0;


}
```

8. Write a program to implement DFA such that it accepts the string that contains substring 001, over alphabet Σ={0,1}.

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

char input[999];
int NFA(int, char);
int main(){
char state;
printf("Enter the string to be checked [INPUTS:0,1] :: ");
scanf("%s",input);

state=NFA(0, 'A');

printf("\n");
if(state=='D')
printf("-->The string is accepted over NFA.\n");
else
printf("-->The string is not accepted over NFA.\n");
return 0;
```

```
}
int NFA(int index,char state){
if(input[index]=='\0'){
return state;
}
if(state=='A'){
if(input[index]=='0'){
char next_state1=NFA(index+1,'A');
char next_state2=NFA(index+1,'B');
if((next_state1=='D')||(next_state2=='D'))
return 'D';
}
else if(input[index]=='1')
return NFA(index+1,'A');
}
else if(state=='B'){
if(input[index]=='0')
return NFA(index+1,'C');
}
else if(state=='C'){
if(input[index]=='1')
return NFA(index+1,'D');
}
else if(state=='D'){
return NFA(index+1,'D');
}
return 'X';


}
```

9. Write a program to implement a PDA that accepts equal number of 0's and 1's where n ≥ 1 by final state.

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


struct STACK{
char value[100];
int top;
};
struct STACK S;
void PUSH(char);
void POP();
int main(){
char input[100];
char state='A';
int i;
S.top=-1;
printf("Enter the string to checked [INPUTS : {0,1} ] :: ");
scanf("%s",input);
```

```
if(state=='A'){
PUSH('Z');
state='B';
}
for(i=0;input[i]!='\0';i++){
char ch=S.value[S.top];
if(state=='B'){
if(input[i]=='0'){
if(ch=='Z')
PUSH('0');
else if(ch=='1')
POP();
else if(ch=='0')
PUSH('0');
}
else if(input[i]=='1'){
if(ch=='Z')
PUSH('1');
else if(ch=='0')
POP();
else if(ch=='1')
PUSH('1');
}
}

}
if(S.value[S.top]=='Z')
state='C';
if(state=='C')
printf("-->The string is accepted over PDA.\n");
else
printf("-->The string is not accepted over PDA.\n");
return 0;
}
void PUSH(char ch){
S.top++;
S.value[S.top]=ch;
}
void POP(){
S.top--;
}
```

10. Write a program to implement a PDA that accepts equal number of 0's and 1's where n ≥ 1 by empty stack.

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

struct STACK{
char value[100];
int top;
```

```
};
struct STACK S;
void PUSH(char);
void POP();
int main(){
char input[100];
char state='A';
int i;
S.top=-1;
printf("Enter the string to checked [INPUTS : {0,1} ] :: ");
scanf("%s",input);

PUSH('E');
for(i=0;input[i]!='\0';i++){
char ch=S.value[S.top];

if(state=='A'){
if(ch=='E'){
PUSH('Z');
i--;
}

else{
if(input[i]=='0'){
if(ch=='Z')
PUSH('0');
else if(ch=='1')
POP();
else if(ch=='0')
PUSH('0');
}
else if(input[i]=='1'){
if(ch=='Z')
PUSH('1');
else if(ch=='0')
POP();
else if(ch=='1')
PUSH('1');
}
}

}

}
if(S.value[S.top]=='Z'){
POP();
}
if(S.value[S.top]=='E')
printf("-->The string is accepted over PDA.\n");
else
printf("-->The string is not accepted over PDA.\n");
return 0;
```

```
}
void PUSH(char ch){
S.top++;
S.value[S.top]=ch;
}
void POP(){
S.top--;
}
```

11. Write a program to implement a PDA that accepts 0_n1_n , where n ≥ 1 by final state.

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

struct STACK{
char value[100];
int top;
};
struct STACK S;
void PUSH(char);
void POP();
int main(){
char input[100];
char state='A';
int i;
S.top=-1;
printf("Enter the string to checked [INPUTS : {0,1} ] :: ");
scanf("%s",input);

PUSH('Z');

for(i=0;input[i]!='\0';i++){
if(state=='A'){
if(input[i]=='0')
PUSH('0');

else if(input[i]=='1'){
POP();
state='B';
}

}
else if(state=='B'){
if(input[i]=='1')
POP();
}
}
```

```
if(S.value[S.top]=='Z')
state='C';
if(state=='C')
printf("-->The string is accepted over PDA.\n");
else
printf("-->The string is not accepted over PDA.\n");
return 0;
}
void PUSH(char ch){
S.top++;
S.value[S.top]=ch;
}
void POP(){
S.top--;
}
```

12. WAP to implement Turing Machine that accepts the language $0^n 1^n$ Where n >= 1.

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

int main(){
char input[100];
char state='A';
int i=0,n;
printf("Enter the string to checked [INPUTS : {0,1} ] :: ");
scanf("%s",input);

n=strlen(input);
input[n]='B';
input[n+1]='\0';

while(input[i]!='\0'){
switch(state){
case 'A':
if (input[i]=='0'){
input[i]='X';
state='B';
i++;
}
else if(input[i]=='Y'){
state='D';
i++;
}
else{
state='R';
}
break;
```

```
case 'B':
if (input[i]=='0'){
i++;
}
else if(input[i]=='Y'){
i++;
}
else if(input[i]=='1'){
input[i]='Y';
state='C';
i--;
}
else {
state='R';
}
break;

case 'C':
if (input[i]=='0'){
i--;
}
else if(input[i]=='Y'){
i--;
}
else if(input[i]=='X'){
state='A';
i++;
}
else {
state='R';
}
break;

case 'D':
if (input[i]=='Y'){
i++;
}
else if(input[i]=='B'){
state='E';
i++;
}
else {
state='R';
}
break;
case 'R':
goto end;
}
}
end:
if(state=='E')
printf("-->The string is accepted over Turing Machine.\n");
```

```
else
printf("-->The string is not accepted over Turing Machine.\n");
return 0;
}
```