COMPILER DESIGN LAB

1) Write c program for DFA accepting the language containing even binary numbers

```
main()

 {

int input[10],i,n;

printf("enter lenth of input" );

scanf("%d",&n);

printf("enter input");

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

        scanf("%d",&input[i]);

    ch=0;

     i=0;

        do
        {
        switch(i)
        {
        case 0: if(input[i]==0)
                ch=1;

        else

                ch=0;
                break;
        case 1: if(input[i]==0)
                ch=1;
                else
                ch=0;
                break;

    }

    i++;

        }

    while(i<n);

        if(ch==1)
        printf("entered number is even binary number");
else
printf("input is not even binary number");
getch();
}
```

OUTPUT:

```
enter length of input 3

enter input 1 1 0

 entered number is even binary number
```

2. Implement DFA that accept all the strings of a’s and b’s 3 rd symbol from is RHS always a.

```c
#include <stdio.h>

#include <string.h>

int isAccepted(char input[]) {

   int len = strlen(input);

   if (len < 3) {

      return 0;

   }

   if (input[len - 3] == 'a') {

      for (int i = 0; i < len - 3; i++) {

         if (input[i] != 'a' && input[i] != 'b') {

            return 0;

         }

      }

      return 1;

   } else {

      return 0;

   }

}


int main() {

   char input[100];

   printf("Enter a string of 'a's and 'b's: ");

   scanf("%s", input);

   if (isAccepted(input)) {

      printf("String accepted!\n");

   } else {
        printf("String rejected!\n");
    }
    return 0;
}
```

OUTPUT:

```
Enter a string of 'a's and 'b's: abbbb

String rejected!

Enter a string of 'a's and 'b's: ababb

String accepted!

Enter a string of 'a's and 'b's: aaaaaa

String accepted!
```

3. **Implement DFA accepting the language of strings not ending with 00 over the input (0,1)**

```
main()

{

int input[10],i,n;

printf("enter lenth of input" );

scanf("%d",&n);

printf("enter input");

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

scanf("%d",&input[i]);

ch=0;

i=0;

do

{

switch(i)

{

case 0: if(input[i]==0)

            ch=1;

            else

            ch=0;

            break;

case 1: if(input[i]==0)

            ch=2;

            else

            ch=0;

            break;

case 2: if(input[i]==0)

            ch=2;
            else
            ch=0;
            break;
}
i++;
}
while(i<n);
if(ch==2)
printf("input accepted");
else
printf("input not accepted");
getch();
}
```

OUTPUT:

```
enter length of input 5
 enter input  1 0 1 0 0
 input accepted
```

4. **Implement the DFA that accept all the string of a's and b's where number of a 's is divisible by 3 and number of b's is divisible by 2.**

```
#include <stdio.h>
#include <stdbool.h>
bool isAccepted(const char *input) {
    int state = 0;
    int countA = 0;
    int countB = 0;
    for (int i = 0; input[i] != '\0'; ++i) {
        char symbol = input[i];
        switch (state) {
            case 0:
                if (symbol == 'a') {
                    state = 1;
                    countA++;
                } else if (symbol == 'b') {
                    state = 2;
                    countB++;
                } else {
                    return false;
                }
                break;

            case 1:
                if (symbol == 'a') {
                    state = 2;
                    countA++;
                } else if (symbol == 'b') {
                    state = 1;
                    countB++;
                } else {
                    return false;  /
                }
                break;

            case 2:
                if (symbol == 'a') {
                    state = 0;
                    countA++;
                } else if (symbol == 'b') {
                    state = 2;
                    countB++;
                } else {
                    return false;
                }
                break;
        }
    }
    return state == 0 && countA % 3 == 0 && countB % 2 == 0;
}

int main() {
    char input[100];
    printf("Enter a string of 'a's and 'b's: ");
    scanf("%s", input);
    if (isAccepted(input)) {
        printf("Accepted\n");
    } else {
        printf("Not accepted\n");
    }

    return 0;
}
```

OUTPUT:

```
Enter a string of 'a's and 'b's: ababa

Accepted
```

5. Write a lex program to count the number of words and number of lines in a given file or program.

```
%{
#include <stdio.h>
int wordCount = 0;
int lineCount = 0;
%}

%%
[a-zA-Z]+   { wordCount++; }
\n          { lineCount++; }
.         { /* ignore other characters */ }

%%

int main() {
   yylex();
   printf("Number of words: %d\n", wordCount);
   printf("Number of lines: %d\n", lineCount);
   return 0;
}
```

6. write recursive descent parser for the grammar E->E+T ,E->T, T->T*F ,T->F ,F->(E)/id.

```
#include<stdio.h>

#include<conio.h>

#include<string.h>

char input[100];

int i,l;

main()

{

  clrscr();

  printf("\n The RDP grammer is\n");

  printf("E->TEP \n EP->+TEP |@ \n T->FTP \n TP->*FTP | @ \n F->(E) | id");

  printf("\n Enter the string to be parsed:");

  scanf("\n %s",input);

  i=0;

  l=strlen(input);

  if(E())

  {

      if(input[i+1]=='\0')

        printf("\nString is accepted");

     else

       printf("\nString is not accepted");

  }

 else

            printf("\n string is rejected");

 getch();

}

E()
```

```
{
  if(T())
  {
  if(EP())
    return 1;
  else
    return 0;
  }
  else
    return 0;
}
EP()
{
  if(input[i]=='+')
  {
    i++;
    if(T())
     {
          if(EP())
        return 1;
      else
        return 0;
     }
  else
    return 0;
 }
  else
```

```
      return 1;
}
T()
{
  if(F())
   {
     if(TP())
       return 1;
     else
       return 0;
   }
 else
    return 0;
}
TP()
{
   if(input[i]=='*')
   {
      i++;
      if(F())
       {
                 if(TP())
             return 1;
           else
              return 0;
       }
     else
```

```
      return 0;
   }
else
  return 1;
}
F()
{
   if(input[i]=='(')
   {
      i++;
   if(E())
   {
      if(input[i]==')')
      {
        i++;
         return 1;
      }
    else
        return 0;
   }
   }
 else if(input[i]=='i')
{
   i++;
   if(input[i]=='d')
   {
     i++;
    return 1;
   }
}
else
  return 0;
}
```

OUTPUT:

```
E->TEP
EP->+TEP | ϵ
T->FTP
TP->*FTP | ϵ
 F->(E) | id
Enter the string to be parsed id+id*id
String is accepted
```

7. write recursive descent parser for the grammar S->(L) S->a L->L,S L->S

```
#include<stdio.h>

#include<conio.h>

#include<string.h>

char input[100];

int i,l;

main()

{

  clrscr();

  printf("\n The RDP grammer is\n");

  printf("\n S->(L)  \n  S->a  \n  L->L,S   \n  L->S");

  printf("\n Enter the string to be parsed:");

  scanf("\n %s",input);

  i=0;

  l=strlen(input);

  if(S())

  {

      if(input[i+1]=='\0')

        printf("\nString is accepted");

     else

       printf("\nString is not accepted");

  }

 else

           printf("\n string is rejected");

 getch();

}
```

```
L()
{
  if(S())
  {
  if(LP())
    return 1;
  else
    return 0;
  }
  else
    return 0;
}
LP()
{
  if(input[i]==',')
  {
    i++;
    if(S())
     {
           if(LP())
        return 1;
      else
        return 0;
     }
  else
    return 0;
  }
   else
      return 1;
}

S()
{
   if(input[i]=='(')
   {
      i++;
   if(L())
   {
      if(input[i]==')')
      {
        i++;
         return 1;
      }
    else
        return 0;
   }
   }
 else if(input[i]=='a')
{
   i++;
}
else
```

```
  return 0;

}
```

OUTPUT:

```
S->(L)

S->a

L->L,S

L->S

Enter the string to be parsed (a,(a,a))

String is accepted
```

8. **Write a C program to calculate first function for the grammar**

E->E+T ,E->T ,T->T*F, T->F,F->(E)/id

```
#include <stdio.h>
#include <ctype.h>
void E();
void T();
void F();
const char *input;
char getNextToken() {
  return *input++;
}
void printFirstSet(char nonTerminal, char firstSet[]) {
  printf("First(%c) = {", nonTerminal);
  for (int i = 0; firstSet[i] != '\0'; ++i) {
    printf(" %c", firstSet[i]);
    if (firstSet[i + 1] != '\0') {
      printf(",");
    }
  }
  printf(" }\n");
}
void addToFirstSet(char firstSet[], char symbol) {
  for (int i = 0; firstSet[i] != '\0'; ++i) {
    if (firstSet[i] == symbol) {
      return;
    }
  }
  firstSet[strlen(firstSet)] = symbol;
  firstSet[strlen(firstSet) + 1] = '\0';
}
void first_E(char firstSet[]);
void first_T(char firstSet[]) {
  char token = getNextToken();

  if (isalpha(token) || token == '(' || token == 'id') {
    F(firstSet);
    T(firstSet);
  }
```

```
}
void first_F(char firstSet[]) {
    char token = getNextToken();

    if (isalpha(token) || token == '(' || token == 'id') {
        E(firstSet);
    }
}
void first_E(char firstSet[]) {
    char token = getNextToken();

    if (isalpha(token) || token == '(' || token == 'id') {
        T(firstSet);
        E(firstSet);
    }
}

int main() {
    const char *inputString = "id+id*id";
    input = inputString;

    char firstSetE[100] = "";
    first_E(firstSetE);
    printFirstSet('E', firstSetE);

    return 0;
}
```

OUTPUT:

```
First(E) = { id, ( }
```