**Time: 1 Hour**

**Answer only 1 from below alternatively according to your seating order**

**Question 1**

**NAME OF THE EXPERIMENT:** Predictive Parser of the **RECURSIVE DESCENT PARSER**

S -> ABabCDef

A -> Bbc

B -> Cbcd

C -> cdef

D -> d

**Input String: cdefbcdbccdefbcdabcdefdef or any other string accepted by this grammer**

**THEORY**

Recursive descent parsing is probably the most well-known and intuitive technique applicable to a subclass of context-free grammars. A subroutine for each non-terminal should determine a rule according to which the substring shall be parsed.

**ALGORITHM:**

**Step1:** Start

**Step2:** Declaration of the functions int S(),int A(),int B(),int C(),int D();

**Step3:** Declare the variable char input[100];int i;

**Step4:** Enter the string

**Step5:** Read the by gets(input);

**Step6:** The given input is true when (S()==1)

**Step7:** Print string is accepted

**Step8:** Otherwise print string is not accepted

**Step 9:** called method int S() j{

if(A()==1)

if(B()==1)

if(input[i]=='a')

{

Increment i; if(input[i]=='b')

{

Increment i; if(C()==1) if(D()==1) if(input[i]=='e')

{

Increment i; if(input[i]=='f')

{

Increment i; return 1;

}

else return 0;

}

else return 0; else return 0;

else return 0;

}

else return 0;

}

else return 0; else return 0;

else return 0;

}

**Step10:** called method int A()

{

if(B()==1)

if(input[i]=='b')

{

Increment i; if(input[i]=='c')

{

Increment i; return 1;

}

else return 0;

}

else return 0; else return 0;

}

**Step11:** called method int B()

{

if(C()==1)

if(input[i]=='b')

{

Increment i; if(input[i]=='c')

{

Increment i; if(input[i]=='d')

{

Increment i; return 1;

}

else return 0;

}

else return 0;

}

else return 0; else return 0;

}

**Step12:** Called method int C()

{

if(input[i]=='c')

{

Increment i; if(input[i]=='d')

{

Increment i; if(input[i]=='e')

{

Increment i; if(input[i]=='f')

{

Increment i; return 1;

}

else return 0;

}

else return 0;

}

else return 0;

}

else return 0;

}

**Step13:** Called method int D()

{

if(input[i]=='d')

{

Increment i;

|  |  |
| --- | --- |
| return 1; | |
| } | |
| else return 0; | |
| } | |
| **Step14:** | Stop |

**Question 2**

\*Write a C program to recognize strings under 'a\*c\*', 'a\*b+c', 'abb+c'. **3.2 RESOURCE:** Turbo C++ **3.3 PROGRAM LOGIC:** By using transition diagram we verify input of the state. If the state recognize the given pattern rule. Then print string is accepted Else print string not accepted.

while(s[i]!='\0') {

switch(state) {

case 0: c=s[i++];

if(c=='a') state=1;

else if(c=='b') state=2;

else state=6;

break;

case 1: c=s[i++];

if(c=='a')

state=3;

else if(c=='b')

state=4;

else state=6;

break;

case 2: c=s[i++];

if(c=='a') state=6;

else if(c=='b') state=2;

else state=6; break; case 3: c=s[i++];

if(c=='a') state=3;

else if(c=='b') state=2;

else state=6;

break;

case 4: c=s[i++];

if(c=='a') state=6;

else if(c=='b') state=5;

else state=6;

break;

case 5: c=s[i++];

if(c=='a') state=6;

else if(c=='b') state=2;

else state=6;

break;

case 6: printf("\n %s is not recognised.",s); exit(0); } }

if(state==1)

printf("\n %s is accepted under rule 'a'",s);

else if((state==2)||(state==4))

printf("\n %s is accepted under rule 'a\*b+'",s); else if(state==5)

printf("\n %s is accepted under rule 'abb'",s);

getch(); }

**3.6 INPUT & OUTPUT:**

**Input :** Enter a String: aaaabbbbbc

**Output:** aaaabbbbbc is accepted under rule 'a\*b+c'

Enter a string: cdgs

cdgs is not recognized