190905104
Parth Shukla
Lab 5

Write a recursive descent parser for the following simple grammars:

Q1) S -> a | > | (T)
T -> T,S | S

After removing left recursion,

S -> a | > | (T)
T -> ST'
T' -> ,ST' | E

(Note: E is epsilon)

```
/*
S -> a | > | (T)
T -> T,S | S

After removing left recursion,

S -> a | > | (T)
T -> ST'
T' -> ,ST' | E
*/

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

int cur = 0;
char s[1000];

void S();
void T();
void Tprime();

void invalid(){
        printf("----------------ERROR!----------------\n");
        exit(0);
}

void valid(){
        printf("------------------SUCCESS!------------------\n");
        exit(0);
}

void S(){
   if(s[cur] == 'a' || s[cur] == '<'){
      cur++;
```

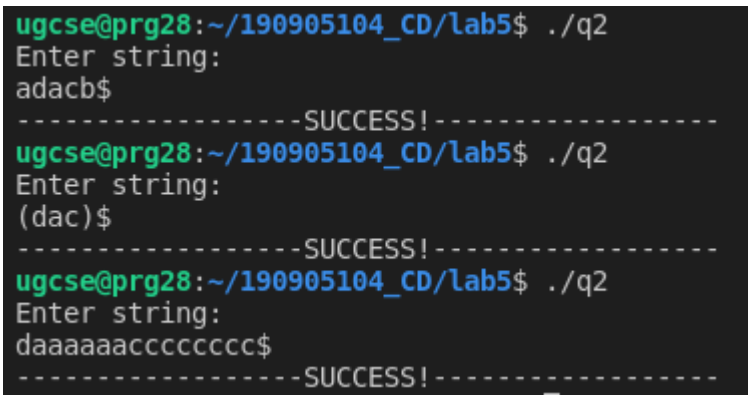
```
        return;
    }
    else if(s[cur] == '('){
        cur++;
        T();
        if(s[cur] == ')'){
            cur++;
            return;
        }
        else{
            invalid();
        }
    }
    else{
        invalid();
    }
}

void T(){
    S();
    Tprime();
}

void Tprime(){
    if(s[cur] == ','){
        cur++;
        S();
        Tprime();
    }
}

void main(){
            printf("Enter string:\n");
            scanf("%s",s);
            S();
            if(s[cur] == '$')
                        valid();
            else
                        invalid();
}
```

```
ugcse@prg28:~/190905104_CD/lab5$ ./q1
Enter string:
(a)$
------------------SUCCESS!------------------
ugcse@prg28:~/190905104_CD/lab5$ ./q1
Enter string:
(a,<,a)$
------------------SUCCESS!------------------
ugcse@prg28:~/190905104_CD/lab5$ ./q1
Enter string:
(a>)$
---------------ERROR!---------------
ugcse@prg28:~/190905104_CD/lab5$ ./q1
Enter string:
a,(a)$
---------------ERROR!---------------
ugcse@prg28:~/190905104_CD/lab5$
```

2)

```
/*
S->UVW
U->(S) | aSb | d
V -> aV | empty
W -> cW | empty
*/

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

int cur = 0;
char s[1000];

void invalid(){
	printf("------------------ERROR!------------------\n");
	exit(0);
}

void valid(){
	printf("------------------SUCCESS!------------------\n");
	exit(0);
}

void S();
void U();
void V();
void W();

void S(){
	U();
	V();
	W();
}
void U(){
	if(s[cur] == '('){
		cur++;
		S();
		if(s[cur] == ')'){
			cur++;
			return;
		}
		else invalid();
	}
	else if(s[cur] == 'd'){
		cur++;
		return;
	}
	else if(s[cur] == 'a'){
		cur++;
```

```
                S();
                if(s[cur] == 'b'){
                        cur++;
                        return;
                }
                else invalid();
        }
        else invalid();
}

void V(){
        if(s[cur] == 'a'){
                cur++;
                V();
        }
}

void W(){
        if(s[cur] == 'c'){
                cur++;
                W();
        }
}

void main(){
        printf("Enter string:\n");
        scanf("%s",s);
        S();
        if(s[cur] == '$')
                valid();
        else
                invalid();
}
```



3)

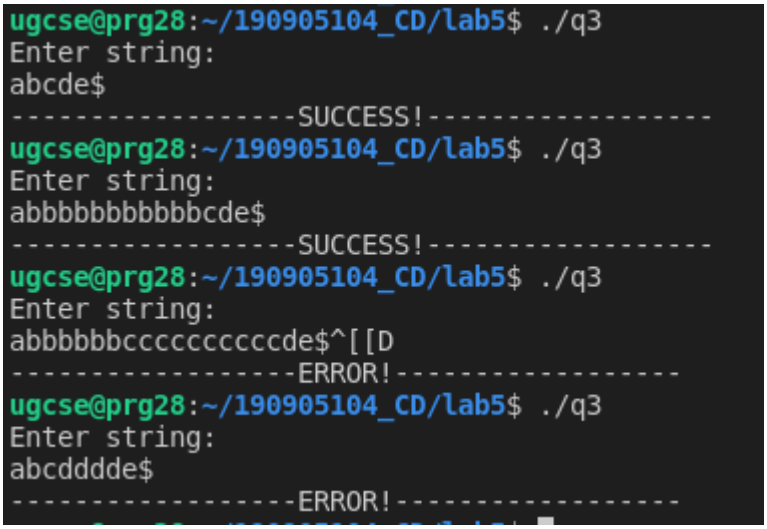
/*

S -> aAcBe
A -> Ab | b
B -> d

After eliminating left recursion,
S -> aAcBe
A -> bA’
A’ -> bA’ | E
B -> d
***/**

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

int cur = 0;
char s[1000];

void S();
void A();
void Aprime();
void B();

void invalid(){
        printf("------------------ERROR!------------------\n");
        exit(0);
}

void valid(){
        printf("------------------SUCCESS!------------------\n");
        exit(0);
}

void S(){
   if(s[cur] == 'a'){
      cur++;
      A();
      if(s[cur] == 'c'){
         cur++;
         B();
         if(s[cur] == 'e'){
            cur++;
            return;
         }
         else{
            invalid();
         }
      }
      else{
         invalid();
      }
   }
```

```
    else{
        invalid();
    }
}

void A(){
    if(s[cur] == 'b'){
        cur++;
        Aprime();
    }
    else{
        invalid();
    }
}

void Aprime(){
    if(s[cur] == 'b'){
        cur++;
        Aprime();
    }
}

void B(){
    if(s[cur] == 'd'){
        cur++;
        return;
    }
    else{
        invalid();
    }
}

void main(){
        printf("Enter string:\n");
        scanf("%s",s);
        S();
        if(s[cur] == '$')
                valid();
        else
                invalid();
}
```



4)

/*S -> (L) | a
L -> L,S | S

After removing left recursion,

S -> (L) | a
L -> SL'
L' -> ,SL' | E
***/**

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

int cur = 0;
char s[1000];

void S();
void L();
void Lprime();

void invalid(){
        printf("------------------ERROR!------------------\n");
        exit(0);
}

void valid(){
        printf("------------------SUCCESS!------------------\n");
        exit(0);
}

void S(){
   if(s[cur] == '('){
      cur++;
      L();
      if(s[cur] == ')'){
         cur++;
         return;
      }
      else{
         invalid();
      }
   }
   else if(s[cur] == 'a'){
      cur++;
      return;
   }
   else{
      invalid();
```

```
    }
}

void L(){
    S();
    Lprime();
}

void Lprime(){
    if(s[cur] == ','){
        cur++;
        S();
        Lprime();
    }
}


void main(){
        printf("Enter string:\n");
        scanf("%s",s);
        S();
        if(s[cur] == '$')
                valid();
        else
                invalid();
}
```