Name: Sapthami Upadhya
Section: CSE A
Roll No. 15
Reg. No.: 230905090

Week 6

Write a recursive descent parser for the following simple grammars.

1. S -> a|>|(T)
   T -> T,S|S

- After removing left recursion,
  S -> a|>|(T)
  T -> ST'
  T' -> ,ST'|ϵ
- Code:

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

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

char str[100];
int cur = 0;

void invalid(){
    printf("---ERROR---\n");
    exit(0);
}
void valid(){
    printf("---SUCCESS---\n");
    exit(0);
}
void S();
void T();
void Tprime();
void S(){
    if(str[cur] == 'a'){
        cur++;
        return;
    }
    else if(str[cur] == '>'){
        cur++;
        return;
```

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

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

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

int main(){
    printf("Enter String: ");
    scanf("%s", str);
    S();
    if(str[cur] == '$') valid();
    else invalid();
}
```

- Output:

```
sapthamiupadhya@Sapthamis-MacBook-Air Lab6 % ./ex1
Enter String: (a,>,(a,>))$
---SUCCESS---
sapthamiupadhya@Sapthamis-MacBook-Air Lab6 % ./ex1
Enter String: (a,>,)$
---ERROR---
```

2. S -> UVW
   U -> (S)|aSb|d
   V -> aV | ϵ
   W -> cW | ϵ

- No left recursion
- Code:

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

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

char str[100];
int cur = 0;

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

```
    else if(str[cur] == 'd'){
        cur++;
        return;
    }
    else invalid();
}

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

void W(){
    if(str[cur] == 'c'){
        cur++;
        W();
    }
}
int main(){
    printf("Enter String: ");
    scanf("%s", str);
    S();
    if(str[cur] == '$') valid();
    else invalid();
}
```

- Output:

```
sapthamiupadhya@Sapthamis-MacBook-Air Lab6 % ./ex2
Enter String: a(d)baacc$
---SUCCESS---
sapthamiupadhya@Sapthamis-MacBook-Air Lab6 % ./ex2
Enter String: a(d)baaccb$
---ERROR---
```

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

- After removing left recursion,
  S -> aAcBe
  A -> bA'
  A' -> bA'|ϵ
  B -> d

- Code:

```
/*
S -> aAcBe
A -> bA'
A' -> bA'|ϵ
B -> d
*/

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

char str[100];
int cur = 0;

void invalid(){
    printf("---ERROR---\n");
    exit(0);
}
void valid(){
    printf("---SUCCESS---\n");
    exit(0);
}
void S();
void A();
void Aprime();
void B();

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

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

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

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


int main(){
    printf("Enter String: ");
    scanf("%s", str);
    S();
    if(str[cur] == '$') valid();
    else invalid();
}
```

- Output:

```
sapthamiupadhya@Sapthamis-MacBook-Air Lab6 % ./ex3
Enter String: abbcde$
---SUCCESS---
sapthamiupadhya@Sapthamis-MacBook-Air Lab6 % ./ex3
Enter String: abbdce
---ERROR---
```

4. S -> (L)|a
   L -> L,S|S

- After removing left recursion,
  S -> (L)|a
  L -> SL'
  L' -> ,SL'|ϵ

- Code:

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

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

```c
#include<string.h>
#include<ctype.h>

char str[100];
int cur = 0;

void invalid(){
    printf("---ERROR---\n");
    exit(0);
}
void valid(){
    printf("---SUCCESS---\n");
    exit(0);
}
void S();
void L();
void Lprime();

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

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

}

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


int main(){
    printf("Enter String: ");
    scanf("%s", str);
    S();
```

```
    if(str[cur] == '$') valid();
    else invalid();
}
```

- Output: