**LAB6 : RECURSIVE DESCENT PARSER FOR SIMPLE GRAMMARS**

Disha Jain – 220905554 – 61

1. **S→a | > | ( T )**

**T→T, S|S**

/\* remove left recursion

S → a | > | ( T )

T → ST'

T' → ,ST' | 𝜖

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int cur = 0;

char str[100];

void invalid()

{

printf("-----------------ERROR!----------------\n");

exit(0);

}

void valid()

{

printf("----------------SUCCESS!---------------\n");

exit(0);

}

void S();

void T();

void Tdash();

void S()

{

if(str[cur]=='a' || str[cur]=='>')

{

cur++;

return;

}

else if(str[cur]=='(')

{

cur++;

T();

if(str[cur]==')')

{

cur++;

return;

}

else

invalid();

}

else

invalid();

}

void T()

{

S();

Tdash();

}

void Tdash()

{

if(str[cur]==',')

{

cur++;

S();

Tdash();

}

return; //this because of epsilon no need to compare any char it is accepted

}

int main()

{

printf("Enter String: ");

scanf("%s", str);

S();

if(str[cur] == '$')

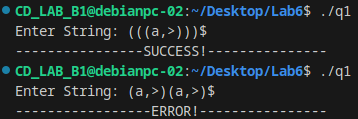
valid();

else

// printf("%c\n", str[curr]);

invalid();

}



1. **S→UVW**

**U →(S) | aSb | d**

**V → aV | 𝜖**

**W → cW | 𝜖**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int cur = 0;

char str[100];

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();

}

return; //epsilon

}

void W()

{

if(str[cur]=='c')

{

cur++;

W();

}

return; //epsilon

}

int main()

{

printf("Enter String: ");

scanf("%s", str);

S();

if(str[cur] == '$')

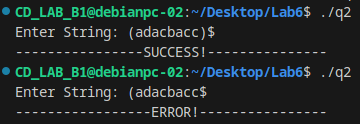
valid();

else

// printf("%c\n", str[curr]);

invalid();

}



1. **S→aAcBe**

**A→Ab|b**

**B→d**

/\* remove left recursion

S → aAcBe

A → bA'

A' → bA' | 𝜖

B → d

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int cur = 0;

char str[100];

void invalid()

{

printf("-----------------ERROR!----------------\n");

exit(0);

}

void valid()

{

printf("----------------SUCCESS!---------------\n");

exit(0);

}

void S();

void A();

void Adash();

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++;

Adash();

}

else

invalid();

}

void Adash()

{

if(str[cur]=='b')

{

cur++;

Adash();

}

return; //epsilon

}

void B()

{

if(str[cur]=='d')

{

cur++;

return;

}

}

int main()

{

printf("Enter String: ");

scanf("%s", str);

S();

if(str[cur] == '$')

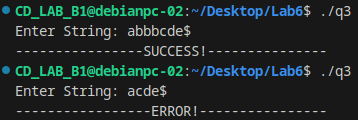
valid();

else

// printf("%c\n", str[curr]);

invalid();

}



**4) S →(L) | a**

**L → L,S | S**

/\* remove left recursion

S → (L) | a

L → SL'

L' → ,SL' | 𝜖

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int cur = 0;

char str[100];

void invalid()

{

printf("-----------------ERROR!----------------\n");

exit(0);

}

void valid()

{

printf("----------------SUCCESS!---------------\n");

exit(0);

}

void S();

void L();

void Ldash();

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();

Ldash();

}

void Ldash()

{

if(str[cur]==',')

{

cur++;

S();

Ldash();

}

return; //epsilon

}

int main()

{

printf("Enter String: ");

scanf("%s", str);

S();

if(str[cur] == '$')

valid();

else

// printf("%c\n", str[curr]);

invalid();

}

