/* B1: Write a program to implement
(a) Recursive Descent Parsing with back tracking (Brute Force Method)

$$S \rightarrow cAd$$
$$A \rightarrow ab / a$$

(b) Recursive Descent Parsing with back tracking (Brute Force Method)

$$S \rightarrow cAd$$
$$A \rightarrow a / ab$$

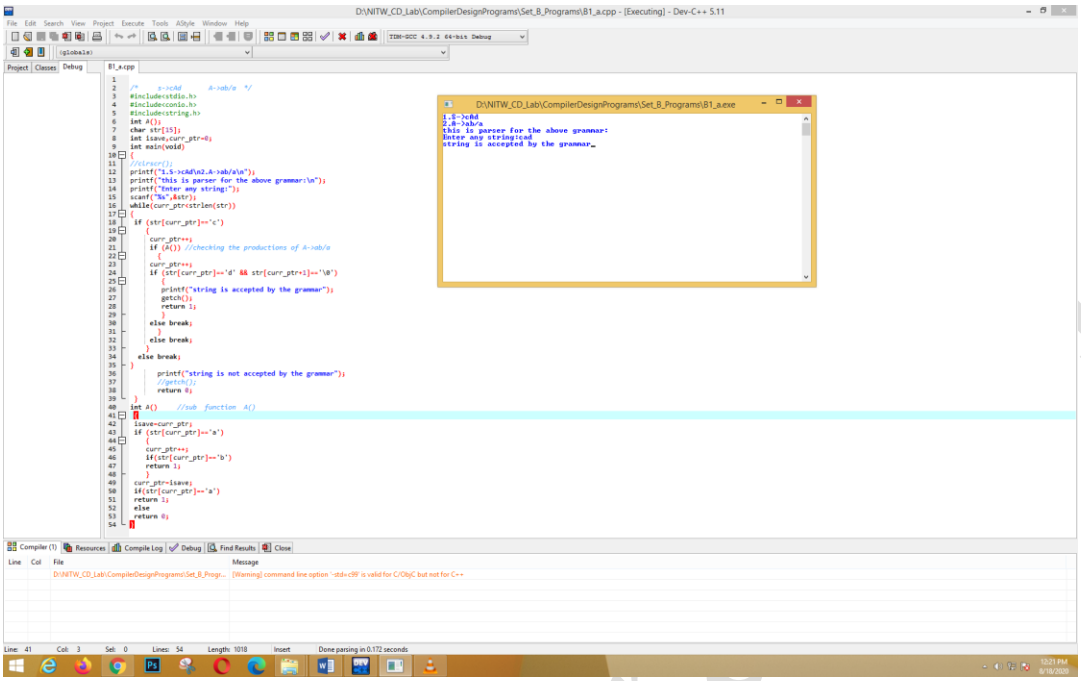
Note: What is the difference that you have experienced w.r.t. the language accepted for these two CFGs.
*/

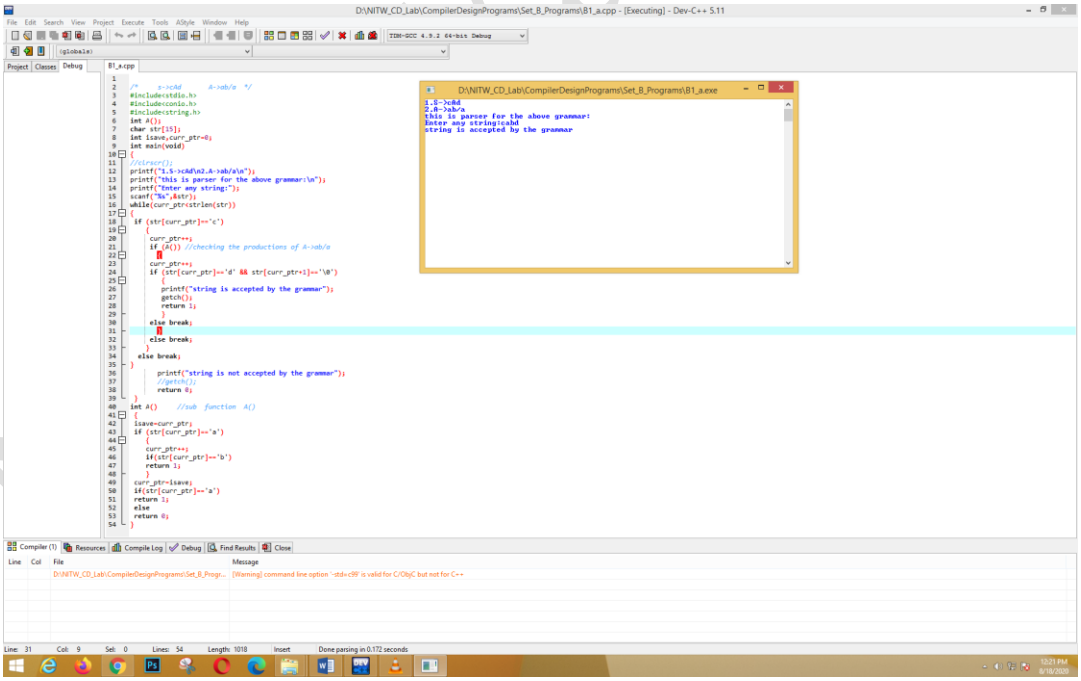
/* s->cAd A->ab/a */

File: B1_a.cpp

```
/*      s->cAd        A->ab/a   */
#include<stdio.h>
#include<conio.h>
#include<string.h>
int A();
char str[15];
int isave,curr_ptr=0;
int main(void)
{
//clrscr();
printf("1.S->cAd\n2.A->ab/a\n");
printf("this is parser for the above grammar:\n");
printf("Enter any string:");
scanf("%s",&str);
while(curr_ptr<strlen(str))
{
//S has only one immediate derivation which is cAd

//match with c
 if (str[curr_ptr]=='c')
    {
     curr_ptr++;
     //call function to match A
     if (A()) //checking the productions of A->ab/a
       {
      curr_ptr++;
     //match d
      if (str[curr_ptr]=='d' && str[curr_ptr+1]=='\0')
         {
     //success
         printf("string is accepted by the grammar");
         getch();
         return 1;
         }
      else break;
```

```
       }
     else break;
    }
  else break;
}
      //incase any of them fail to match return negatively.
       printf("string is not accepted by the grammar");
       //getch();
       return 0;
 }

int A()     //sub  function  A()
 {
//this function matches all terminal strings generated by the variable
A. Here the only terminal strings generated by A is ab or a.
 isave=curr_ptr;

//match with a and advance and match with b. If successful return
positive.
 if (str[curr_ptr]=='a')
    {
    curr_ptr++;
    if(str[curr_ptr]=='b')
    return 1;
    }
 curr_ptr=isave; //return to start

//check if a is matched and return accordingly.
 if(str[curr_ptr]=='a')
 return 1;
 else
 return 0;
}
```


D:\NITW_CD_Lab\CompilerDesignPrograms\Set_B_Programs\B1_a.cpp - [Executing] - Dev-C++ 5.11
B1_a.cpp
D:\NITW_CD_Lab\CompilerDesignPrograms\Set_B_Programs\B1_a.exe
1.S->cAd
2.A->ab/a
this is parser for the above grammar:
Enter any string:cad
string is accepted by the grammar
Compiler (1)
Resources
Compile Log
Debug
Find Results
Close
Done parsing in 0.172 seconds


D:\NITW_CD_Lab\CompilerDesignPrograms\Set_B_Programs\B1_a.cpp - [Executing] - Dev-C++ 5.11
B1_a.cpp
D:\NITW_CD_Lab\CompilerDesignPrograms\Set_B_Programs\B1_a.exe
1.S->cAd
2.A->ab/a
this is parser for the above grammar:
Enter any string:cabd
string is accepted by the grammar
Compiler (1)
Resources
Compile Log
Debug
Find Results
Close
Done parsing in 0.172 seconds

File: B1_b.cpp

```
/*
    The Production are:
                                        S -> cAd
        A -> a/ab
*/

#include <stdio.h>
#include <conio.h>
#include <ctype.h>
int S ();
int A ();
int i=0;
char S1[10];
int main()
{
int j=0;
//clrscr();
printf(" Enter Input String \n ");
gets(S1);
j=S ();
if(j) {
printf( " String is Accepted \n ");
}
else {
printf(" String is Rejected \n" );
}
getch();
}

int S ()
{
//S has only one immediate derivation which is cAd

//match with c
    if(S1[i] == 'c')
    {
          i++;
      //call function to match A
        if (A ())
          {
           //match d
                  if(S1[i] =='d')
                   {
                           i++;
                             if(S1[i] != '\0' && S1[i]!= ' ' &&
S1[i]!= '\t') return 0;
                          else return 1;
                  }
        }
```

```
 }
return 0;
}
int A ()
{
//this function matches all terminal strings generated by the variable
A. Here the only terminal strings generated by A is ab or a.
      //??
      if(S1[i] == 'a')
      {
           i++;
          return 1;
     }
      if(S1[i] == 'a')
    {
      i++;
      if(S1[i]=='b')
     {
        i++;
       return 1;
    }
     return 1;
}
else
return 0;
}
```