Program Number : 03

1BY18CS177 6 C

Taruish J. Reddy

Question

Design, develop and implement yacc/c program to construct Predictive LL(1) parsing Table for the grammar rules:
A→aBa , B→bB | ε . Use this table to parse the sentence : abba $

Answer

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
char prod[3][10]={"A→aBa", "B→bB", "B→B"}, input[10]
                                              stack[2];
int top=-1; int j=0; k, l;
void push (char item)
{
 stack[++top]=item;
 }
void pop()
 {
 top=top-1;
 }
void display()
{
 int j;
 for (j=top; j>0; j--)
   printf("%c", stack[j]) }
```

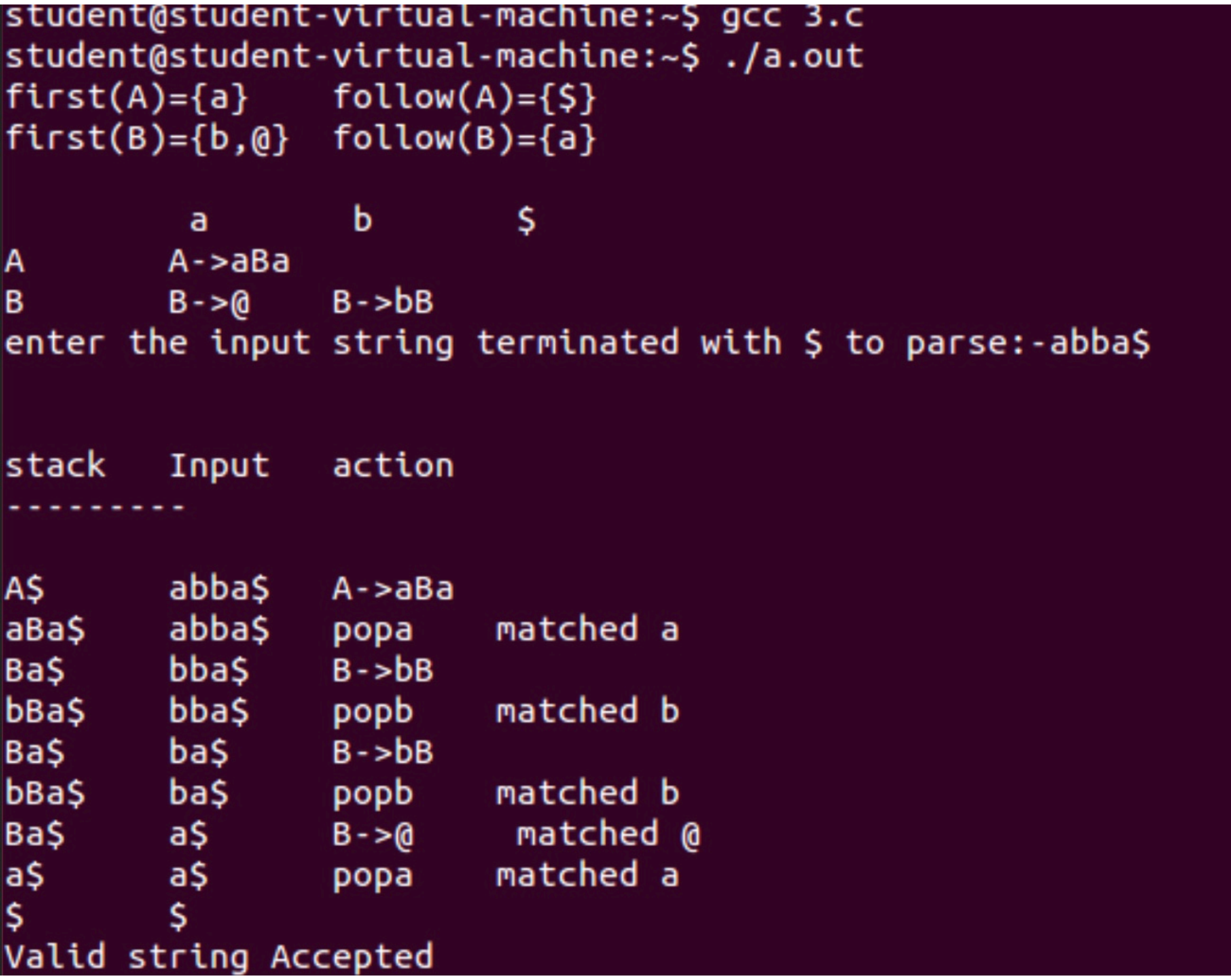
```
void stackpush (char p)
{
 if (p = = 'A')
 { pop();
 for (j = strlen (prod [0] -1 ; j >= 3 ; j--)
        push (prod [0] [j] );
 }
else
 {
 pop();
for (j = strlen (prod [1] - 1 ; j >= 3 ; j--)
        push (prod [1] [j] );
}
 }
void main()
{
  char c; int j;
  printf (" first (A) = {a} \t");
  printf( " follow (A) = {$} \n");
  printf(" first (B) = {b, @} \t");
  printf (" follow (B) = {a} \n\n");
  printf (" \t a \t b \t $ \n");
  printf ("A \t %s \n", prod [0]) ;
  printf ("B \t %s \t %s \n" prod [2] prod [1]);
  printf ("enter the input string terminated with $
  scanf (" %s ", input)
 for (i=0; input [i] != '\0' ; i++)
 {
  if (input [i] != 'a' && input [i] != 'b' && (input [i]
 {
  printf ("invalid string ");
  exit (0);
 } }
```

```
if (input [i-1] ! = '$')
{
printf (" \n \n input sdring entered without and mon ke
exit (0) ;
}
push ('$') ;
push ('A') ;
i = 0
printf ("\n \n stack \t input \t action ");
printf ("\n . . . . . . . . . . \n") ;
while (i != strlen (input) && stack [top] != '$')
{
 printf ('\n')
  for (j = top ; j >= 0 ; j--)
    printf ("%c", stack [j]
  } }
void main ()
{
chan c, int i ;
printf ("finst (A) = {a} \t ");
printf (" follow (A) = {$} \n");
printf (" finst (B) = {b, B} \t");
printf (" follow (B) = {a} \n \n");
printf ("\t a \t b \t $ \n") ;
printf ("A \t %S \n", prod [0]) ;
printf ("B \t %S \t %S \n", prod [2] prod [3]] ;
printf (" enter the input sdring terminated with $ to
scanf ("%S", input)
 for (i = 0; input [i] = \0 ; i++)
```

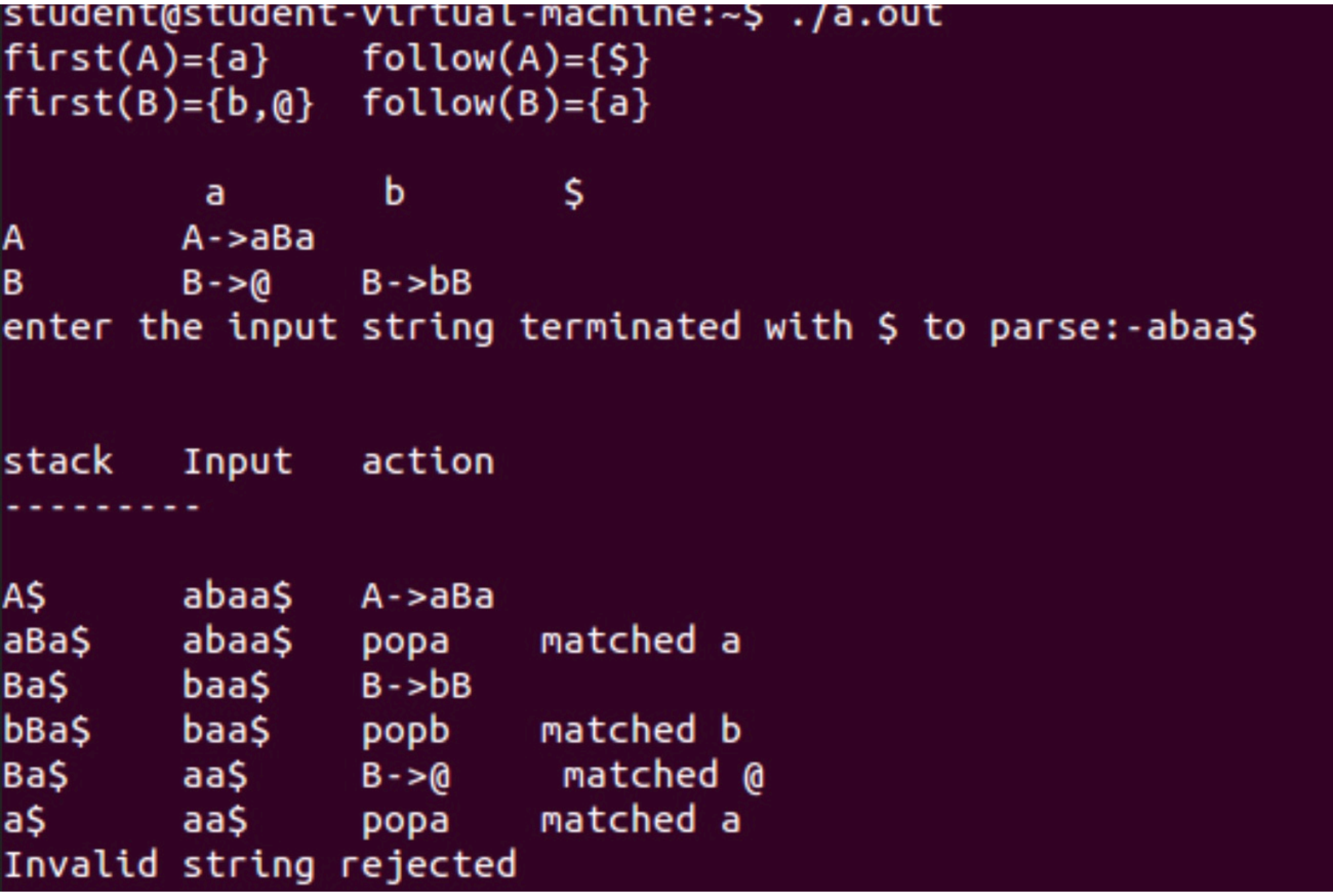
```
if ((input[i] != 'a') && (input[i] != 'b') && (input[i] != '$'))
 {
  printf (" invalid string" )
  exit(0);
 }
}
if (input[i-1] ! = '$')
{
printf (" \n\n Input string entered without end marker
                                                   $");
{ exit(0);
 }
push ('$');
push ('A');
 i = 0
printf (\n\n stack \t input \t action);
printf (\n- - - - - - - - - - - - \n");
while (i ! = s strlen (input) && stack [top] ! = '$')
 {
  printf ("\n")
 for (l = top; l >= 0 ; l--)
  printf ("%c" , stack [l]) ;
  printf ("\t");
 for (l = i; l < strlen (input) ; l++)
       printf ("%c", input [l]);
  printf ("\t");
 if  (stack [top] = = 'A')
 {
   printf ("A→ aBa")
     stack push ("A") ;
 }
else if (stack [top] = = "B")
  { if (input [i] ! = 'B')   {
        printf ("B → @ ")
        printf ("\t matched@ ");
        pop();
    }
   else
   {
   printf ("B → bB")
   stackpush ("B");
   }
 }
 else
 {
  if (stack[top] = = input [i])
  {
   printf( "pop %c", input[i])
   printf ("\t matched %c", input [i]);
   pop();
   i++;
  }
  else
   break;
 }
}
if (stack [top] = = '$' && input [i] = = '$')
{
  printf ("\n $ \t $ ");
  printf ("\n Valid string Accepted \n");
}
else
  printf ("\n Invalid string rejected \n");
}
```

Output

```
$ gcc 3.c
$ ./a.out
first(A) = {a}          follow(A) = {$}
first(B) = {b, @}       follow(B) = {a}.

            a        b         $

A           A → aBa

B           B → @     B → bB

enter the input string determined with $ to parse
abba $

stack       input       action
A$          abba$       A → aBa
aBa $       abba $      pop a        matched a
Ba$         bba $       B → bB
bBa $       bba $       pop b        matched b.
Ba $        ba $        B → bB
bBa $       ba $        pop b        matched b
Ba $        a $         B → @        matched @
a $         a $         pop a        matched a
$           $
Valid string Accepted.
```



```
student@student-virtual-machine:~$ gcc 3.c
student@student-virtual-machine:~$ ./a.out
first(A)={a}    follow(A)={$}
first(B)={b,@}  follow(B)={a}

         a       b       $
A       A->aBa
B       B->@    B->bB
enter the input string terminated with $ to parse:-abba$



stack   Input   action
---------


A$      abba$   A->aBa
aBa$    abba$   popa    matched a
Ba$     bba$    B->bB
bBa$    bba$    popb    matched b
Ba$     ba$     B->bB
bBa$    ba$     popb    matched b
Ba$     a$      B->@     matched @
a$      a$      popa    matched a
$       $
Valid string Accepted
```



```
student@student-virtual-machine:~$ ./a.out
first(A)={a}    follow(A)={$}
first(B)={b,@}  follow(B)={a}

         a       b       $
A       A->aBa
B       B->@    B->bB
enter the input string terminated with $ to parse:-abaa$


stack   Input   action
---------


A$      abaa$   A->aBa
aBa$    abaa$   popa    matched a
Ba$     baa$    B->bB
bBa$    baa$    popb    matched b
Ba$     aa$     B->@     matched @
a$      aa$     popa    matched a
Invalid string rejected
```