```
/* A ->aBa , B ->bB | @*/
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
char prod[3][10]={"A->aBa","B->bB","B->@"};
char first[3][10]={"a","b","@"};
char follow[3][10]={"$","a","a"};
char table[3][3][10];
char input[10];
int top=-1;
char stack[25];
char curp[20];

void push(char item)
{
  stack[++top]=item;
}
void pop()
{
   top=top-1;;
}
void display()
{
   int i;
   for(i=top;i>=0;i--)
      printf("%c",stack[i]);
}

int numr(char c)
{
   switch(c)
    {
    case 'A': return 1;
    case 'B': return 2;
    case 'a': return 1;
    case 'b': return 2;
    case '@': return 3;
     }
return(1);
}

void main()
{
```

```
char c;
int i,j,k,n;
for(i=0;i<3;i++)
for(j=0;j<4;j++)
strcpy(table[i][j],"e");

printf("\n Grammar:\n");
for(i=0;i<3;i++)
        printf("%s\n",prod[i]);

printf("\nfirst= {%s,%s,%s}",first[0],first[1],first[2]);
printf("\nfollow ={%s %s}\n",follow[0],follow[1]);

printf("\nPredictive parsing table for the given grammar\n");
strcpy(table[0][0]," ");
strcpy(table[0][1],"a");
strcpy(table[0][2],"b");
strcpy(table[0][3],"$");
strcpy(table[1][0],"A");
strcpy(table[2][0],"B");

for(i=0;i<3;i++)
{
   k=strlen(first[i]);
   for(j=0;j<k;j++)
   if(first[i][j]!='@')
      strcpy(table[numr(prod[i][0])][numr(first[i][j])],prod[i]);
   else
      strcpy(table[numr(prod[i][0])][numr(follow[i][j])],prod[i]);

}



printf("\n----------------------------------------------------\n");
for(i=0;i<3;i++)
for(j=0;j<4;j++)
{
   printf("%-10s",table[i][j]);
   if(j==3) printf("\n----------------------------------------------------\n");
}

printf("enter the input string terminated with $ to parse :- ");
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\nInput String Entered Without End marker $");
   exit(0);
}
push('$');
push('A');
i=0;
printf("\n\n");
printf(" stack\t   Input  \taction  ");
printf("\n-----------------------------------------------------\n");
while(input[i]!='$' && stack[top]!='$')
{

   display();
   printf("\t\t%s\t ",(input+i));

   if (stack[top]==input[i])
   {
      printf("\tmatched %c\n",input[i]);
      pop();
      i++;
   }
   else
   {
      if(stack[top]>=65 && stack[top]<92)
      {
         strcpy(curp,table[numr(stack[top])][numr(input[i])]);
         if(!(strcmp(curp,"e")))
         {
            printf("\n invalid string- Rejected\n");
            exit(0);
         }
         else
         {
            printf(" \tapply production %s\n",curp);
            if(curp[3]=='@')
               pop();
```

```
                else
                    {
                    pop();
                    n=strlen(curp);
                    for(j=n-1;j>=3;j--)
                    push(curp[j]);

                    }
                }
            }

    }

}
display();
printf("\t\t%s\t ",(input+i));
printf("\n---------------------------------------------------\n");
if(stack[top]=='$' && input[i]=='$' )
    {

        printf("\n valid string - Accepted\n");

    }
else{

            printf("\ninvalid string- Rejected\n");
    }



}
```