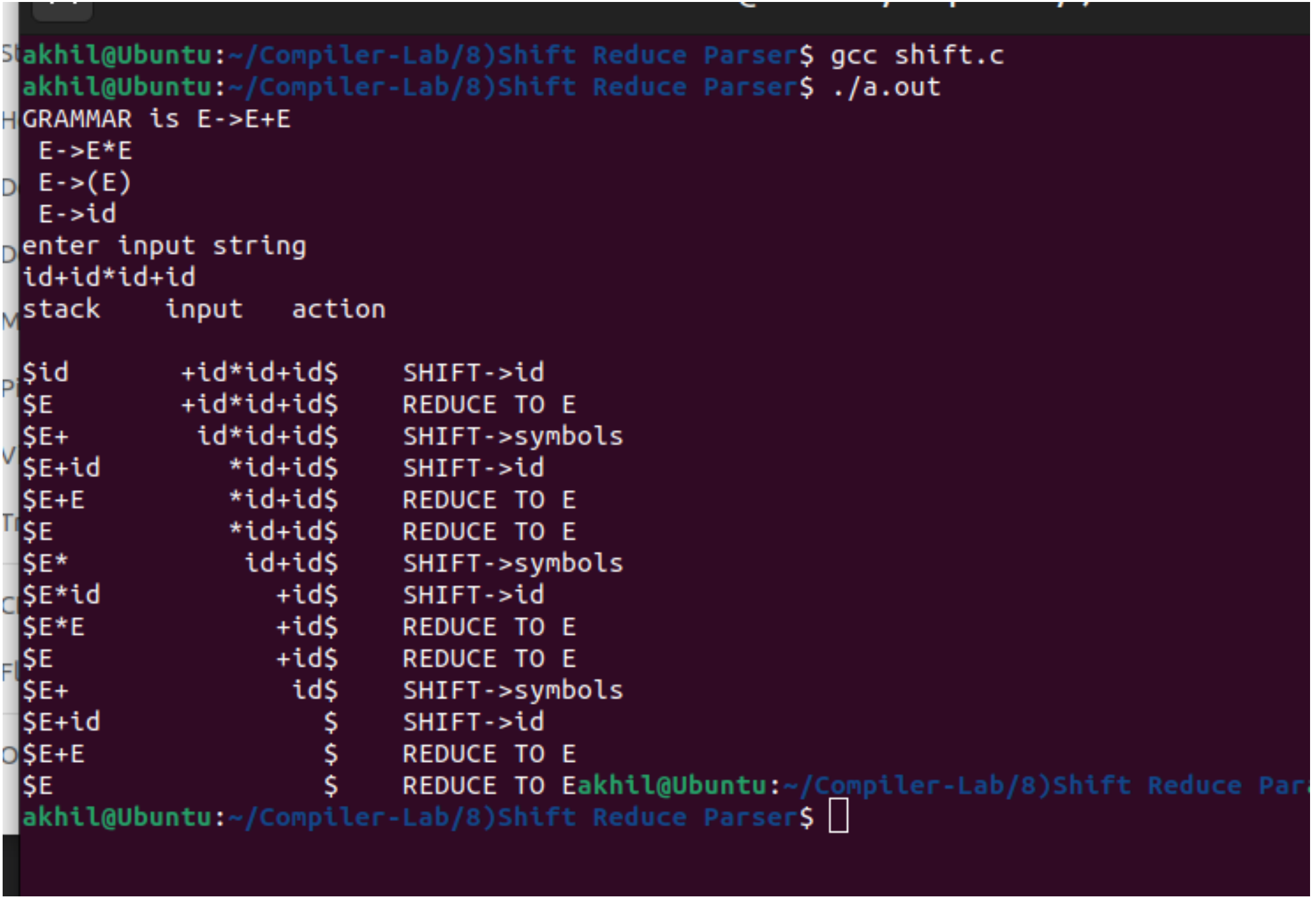
```c
#include<stdio.h>
#include<string.h>
int k=0,z=0,i=0,j=0,c=0;
char a[16],ac[20],stk[15],act[10];
void check();
int main()
   {

      puts("GRAMMAR is E->E+E \n E->E*E \n E->(E) \n E->id");
      puts("enter input string ");
      scanf("%s",a);
      c=strlen(a);
      strcpy(act,"SHIFT->");
      puts("stack \t input \t action");
      for(k=0,i=0; j<c; k++,i++,j++)
       {
         if(a[j]=='i' && a[j+1]=='d')
           {
              stk[i]=a[j];
              stk[i+1]=a[j+1];
              stk[i+2]='\0';
              a[j]=' ';
              a[j+1]=' ';
              printf("\n$%s\t%s$\t%sid",stk,a,act);
              check();
           }
         else
           {
              stk[i]=a[j];
              stk[i+1]='\0';
              a[j]=' ';
              printf("\n$%s\t%s$\t%ssymbols",stk,a,act);
              check();
           }
       }

   }
void check()
   {
     strcpy(ac,"REDUCE TO E");
     for(z=0; z<c; z++)
       if(stk[z]=='i' && stk[z+1]=='d')
         {
           stk[z]='E';
           stk[z+1]='\0';
           printf("\n$%s\t%s$\t%s",stk,a,ac);
           j++;
         }
     for(z=0; z<c; z++)
       if(stk[z]=='E' && stk[z+1]=='+' && stk[z+2]=='E')
         {
           stk[z]='E';
           stk[z+1]='\0';
           stk[z+2]='\0';
           printf("\n$%s\t%s$\t%s",stk,a,ac);
           i=i-2;
         }
     for(z=0; z<c; z++)
       if(stk[z]=='E' && stk[z+1]=='*' && stk[z+2]=='E')
         {
           stk[z]='E';
           stk[z+1]='\0';
           stk[z+1]='\0';
           printf("\n$%s\t%s$\t%s",stk,a,ac);
           i=i-2;
         }
```

```
    for(z=0; z<c; z++)
      if(stk[z]=='(' && stk[z+1]=='E' && stk[z+2]==')')
        {
          stk[z]='E';
          stk[z+1]='\0';
          stk[z+1]='\0';
          printf("\n$%s\t%s$\t%s",stk,a,ac);
          i=i-2;
        }
}
```


akhil@Ubuntu:~/Compiler-Lab/8)Shift Reduce Parser$ gcc shift.c
akhil@Ubuntu:~/Compiler-Lab/8)Shift Reduce Parser$ ./a.out
GRAMMAR is E->E+E
E->E*E
E->(E)
E->id
enter input string
id+id*id+id
stack input action
$id +id*id+id$ SHIFT->id
$E +id*id+id$ REDUCE TO E
$E+ id*id+id$ SHIFT->symbols
$E+id *id+id$ SHIFT->id
$E+E *id+id$ REDUCE TO E
$E *id+id$ REDUCE TO E
$E* id+id$ SHIFT->symbols
$E*id +id$ SHIFT->id
$E*E +id$ REDUCE TO E
$E +id$ REDUCE TO E
$E+ id$ SHIFT->symbols
$E+id $ SHIFT->id
$E+E $ REDUCE TO E
$E $ REDUCE TO Eakhil@Ubuntu:~/Compiler-Lab/8)Shift Reduce Par
akhil@Ubuntu:~/Compiler-Lab/8)Shift Reduce Parser$