

/\*\* Recursive descent parser for the grammar -- O(n) time

S --> aTS | c

T --> bST | c

This parser trace the derivation tree in top-down fashion.

Example

```
      S
     / \
    a  T S
       / | \
      b S T c   abccc is accepted
         | |
         c c
```

This type of parser is possible for a special type of grammar, such as LL(1). \*\*\*/

```
int i,j,k,l,m,n,ii;
char sym;
char input[100];
error(int i){printf("error number %d\n",i);}
getsym(){
    if(i!=0)printf("consume %c ",sym); getchar();
    i++; sym=input[i];
}
S(){
    if(sym=='a'){getsym(); T(); S();}
    else if(sym=='c')getsym();
    else error(1);
    getchar();
}
T(){
    if(sym=='b'){getsym(); S(); T();}
    else if(sym=='c')getsym();
    else error(2);
    getchar();
}
```

```
main() {
    printf("input n "); scanf("%d", &n); getchar();
    for(ii=1;ii<=n;ii++)scanf("%c",&input[ii]);
    for(ii=1;ii<=n;ii++)printf("%c ",input[ii]);
    printf("\n");
    getchar();
    i=0; getsym();
    S();
    if(i==n+1)printf("accept\n");
    else { error(3); printf("reject\n");}
}
/* error 1: a or c expected
   error 2: b or c expected
   error 3: ! expected
*/
```