FX0006: 자료구조 숙제

상명대학교 전기전자컴퓨터학부

숙제 번호	05	점수	/20
제출 날짜	2018년 5월 1일	숙제 디렉토리	~/ds/hw05
학생 이름		학번	

문제 번호	1	점수	10
문제 유형	프로그래밍	프로그램 파일	queue.c

문제 설명

Template 파일 queue.h에 정의된 매크로 및 QUEUE 형을 사용하여 자연수의 queue를 array를 사용하여 처리하는 아래 함수를 파일 queue.c에 구현하라.

① void init_queue(QUEUE * q): queue q를 초기화 한다.

② void print_queue(QUEUE * q): queue q에 저장된 데이터를 출력한다. 출력 형식은 (f|r|n: i1, i2, ..., in)이다. 여기서 f는 front index, r은 rear index, n은 queue q에 저장된 item의 개수이고, i1, i2, ..., in은 front에서 rear 순서로 각 item이다.

③ int empty_queue(QUEUE * q): queue q가 empty이면 1을 그렇지 않으면 0을 return한다.

④ int full_queue(QUEUE * q): queue q가 full이면 1을 그렇지 않으면 0을 return 한다.

⑤ int en_queue(QUEUE * q, int item): queue q에 item을 추가한 후 변화된 rear 값을 return한다. (만약 오류인 경우 ERROR를 return함)

⑥ int de_queue(QUEUE * q): queue q에서 하나의 item을 제거한 후 return한다. (만약 오류인 경우 ERROR를 return함)

시험 프로그램 수행 보기

```
$ test-queue
===== test-queue =====
(0|0|0:)
(0|1|1:10)
(0|2|2:10,11)
(1|2|1:11)
(1|3|2:11,12)
(1|4|3:11,12,13)
(2|4|2:12,13)
(2|5|3:12,13,14)
(2|6|4:12,13,14,15)
(3|6|3:13,14,15)
(3|7|4:13,14,15,16)
(3|0|5:13,14,15,16,17)
(4|0|4:14,15,16,17)
(4|1|5:14,15,16,17,18)
...
OK: full_queue
(7|5|6:17,18,19,20,21,22)
(0|5|5:18,19,20,21,22)
(0|6|6:18,19,20,21,22,24)
(1|6|5:19,20,21,22,24)
(2|6|4:20,21,22,24)
(2|7|5:20,21,22,24,25)
(3|7|4:21,22,24,25)
(4|7|3:22,24,25)
(4|0|4:22,24,25,26)
(5|0|3:24,25,26)
(6|0|2:25,26)
(6|1|3:25,26,27)
(7|1|2:26,27)
...
OK: empty_queue
```

제출물

1.1. 파일 queue.c에서 학생이 작성한 함수를 연필로 쓰고 설명하기. (6점)

1.2. 시험 프로그램 test-queue을 수행한 화면 출력하기. (4점)

1.3. oak 서버의 숙제 디렉토리에 저장된 프로그램 파일.

<table>
<tr><td>문제 번호</td><td>2</td><td>점수</td><td>10</td></tr>
<tr><td>문제 유형</td><td>프로그래밍</td><td>프로그램 파일</td><td>list.c</td></tr>
<tr><td>문제 설명</td><td colspan="3">Template 파일 list.h에 정의된 매크로 및 NODE 형을 사용하여 자연수의 list를 처리하는 아래 함수를 파일 list.c에 구현하라.
① void print_list(NODE * list): list에 포함된 data을 순서대로 출력한다. 출력 형식은 (n: i1, i2, ..., in)이다. 여기서 n은 list에 포함된 node의 개수이고 i1, i2, ..., in은 각 data이다.
② int length_list(NODE * list): list에 포함된 node의 개수를 return한다.
③ NODE *nth_list(NODE * list, int n): list의 n번째(n>=1) node의 주소를 return 한다. (오류 발생 시 NULL을 return)
④ NODE *add_list(NODE * list, int item): list의 맨 앞에 item이 저장된 새 node를 생성하여 추가한 후 수정된 list의 주소를 return한다.
⑤ NODE *delete_list(NODE * list): list의 맨 앞 node를 제거한 후 수정된 list의 주소를 return한다. (오류 발생 시 NULL를 return)</td></tr>
<tr><td rowspan="3">시험
프로그램
수행 보기</td><td><pre>===== test-list0: print_list =====
(0:)
(1:10)
(2:10,20)
...</pre></td><td colspan="2"><pre>===== test-list1: length_list =====
(0:)
(1:10)
(2:10,20)
...</pre></td></tr>
<tr><td><pre>===== test-list2: nth_list =====
OK: nth_list
OK: nth_list
...</pre></td><td colspan="2"><pre>===== test-list3: add_list =====
(0:)
(1:40)
(2:30,40)
...</pre></td></tr>
<tr><td><pre>===== test-list4: delete_list =====
(4:20,30,40,50)
(3:30,40,50)
(2:40,50)
...</pre></td><td colspan="2"></td></tr>
<tr><td>제출물</td><td colspan="3">2.1. 파일 list.c에서 학생이 작성한 함수를 연필로 쓰고 설명하기. (5점)
2.2. 시험 프로그램 test-list0 ~ test-list4를 수행한 화면 출력하기. (5점)
2.3. oak 서버의 숙제 디렉토리에 저장된 프로그램 파일.</td></tr>
</table>

주의: 숙제 디렉토리에 학생이 작성한 프로그램 파일(수행되지 않더라도)이 없는 경우 0점 처리함. 숙제 제출 시간 이 후 프로그램 파일을 수정하는 경우 감점 30% 있음.

끝.