Data Structure: Final Exam

블랙보드에서 기말고사 파일을 다운받아 작성 후 학번_이름.zip 파일로 제출해주세요.

프로젝트 전체 파일 제출하지 마시고 소스코드(.c)와 헤더파일(.h) 만 제출 바랍니다.

ZOOM을 이용해 자신의 화면을 녹화 후 블랙보드에 제출 바랍니다.
(미제출 시 불이익이 있을 수 있습니다.)

* 블랙보드에서 기말고사 자료 다운로드, 코드 제출, ZOOM 실행 이외에는 인터넷 사용이 금지됩니다.

* 카카오톡 등의 메신저는 반드시 로그아웃하고 프로그램을 종료하시기 바랍니다.
(로그인 상태 시 부정행위로 간주됩니다.)

시험 주의 사항

1. 시험 코드는 모두 C 언어로 작성해야 하며, 채점 환경은 기본 실습진행환경 (C Language + Visual Studio 2019 + Windows 10)과 동일합니다.
2. 각각의 문제는 부분점수가 없습니다.
3. 기본적으로 컴파일이 되어야 하며, 주어진 main 문 내의 테스트 케이스(test cases) 이외에도 다른 테스트 케이스를 추가 사용하여 제출하신 코드를 채점합니다.
   A. 컴파일 오류 시 오답처리 됩니다.
   B. 시험자의 환경에 따라 일부 컴파일러에서 문법 오류를 무시하고 관대하게 컴파일이 진행되더라도 채점 환경 기준으로 컴파일 오류를 검사하니 세세한 문법 오류를 주의하시기 바랍니다.
4. 표시된 TODO 영역 이외에 다른 코드는 수정할 수 없습니다.

반드시 블랙보드의 '기말고사 시험 환경 설정 가이드'를 참고하여
시험에 응시하시기 바랍니다.

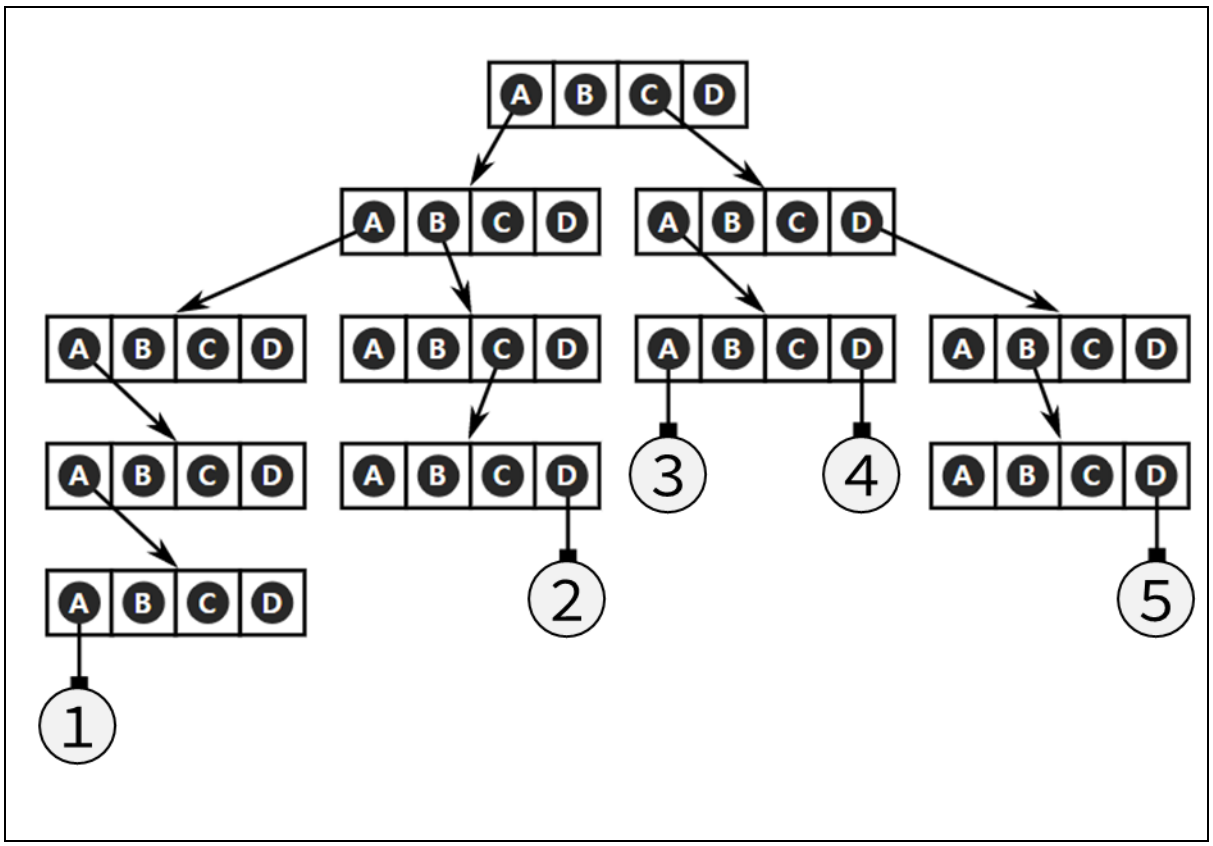
컴퓨터 ZOOM 과 핸드폰 ZOOM 모두 접속하셔야 출석 인정됩니다.
녹화된 컴퓨터 화면은 시험 종료 후
2 시간 이내로 블랙보드 통해 제출바랍니다.

1. 주어진 trie를 보고 다음 질문에 답하시오. (10점)

   A. Figure 1의 1~5번에 해당하는 글자를 작성하시오. (각 2점)

      i. 1_trie.txt 파일 내에 주어진 '출력 예시'와 같이 작성하시오.

Figure 1 Trie



출력예시

```
1. cab
2. cat
3. cow
4. ham
5. hat
```

2. 주어진 조건을 만족하는 이진 탐색 트리 ADT (binary search tree ADT)를 구현하시오. (30 점)

A. Binary search tree는 다음의 조건을 만족함

i. Binary search tree 에는 학생들의 성적 구조체(STUDENT_GRADE)가 담기며, 구조체의 형태는 다음과 같다.

```
typedef struct
{
    studentId id;              // student ID
    char name[20];             // student Name
    int score;                 // student's score
} STUDENT_GRADE;
```

Binary search tree 의 각 노드(node)에는 학생의 학번(id), 이름(name), 성적(score)이 담겨있다.
자세한 사항은 bstADT.h 파일 참조할 것.

ii. Binary search tree는 성적 구조체에서 성적을 키(key)로 사용한다.

iii. 본 binary search tree는 키의 중복을 허용한다.

B. 다음의 함수들을 구현하시오.

- itemInsert (10 점)
  - Binary search tree 에 학생들의 성적 구조체를 삽입하는 함수
    - 새로운 학생의 성적이 현재 노드에 있는 학생의 성적보다 작으면 좌측 노드에 삽입(new < node)
    - 새로운 학생의 성적이 현재 노드에 있는 학생의 성적보다 크거나 같으면 우측 노드에 삽입(new ≥ node)
- itemDelete (10 점)
  - Binary search tree 에서 주어진 성적을 가지는 학생을 "모두" 삭제 하는 함수
    - 예) 점수가 90 점인 학생이 3 명 있을 경우 3 명 모두 삭제
  - 삭제 시 노드의 이동 방향은 마음대로 구현 가능
  - 삭제 실행 결과와 '출력 예시' 내 결과를 비교해볼 것
- itemSearch (10 점)
  - Binary search tree 에서 주어진 성적을 가지는 첫 번째 학생을 검색하는 함수
    - 이진 탐색 트리를 root 부터 탐색할 때 가장 먼저 발견되는 학생을 반환
  - 값을 발견하지 못할 경우 NULL 반환

i. 이 외에도 Binary search tree ADT에서 "TODO" 영역이 있는 나머지 함수 모두를 구현할 것.

ii. 필요에 따라 함수 추가 가능.

1. itemInsert, itemDelete, itemSearch 를 위한 내부 함수 추가 등

출력예시

```
####################
# Insertion Test
PASS
PASS
PASS
PASS
PASS
PASS
PASS
PASS
PASS
PASS
PASS
----------
Node nums: 11
Lee(1010)'s score: 50
Hahn(1003)'s score: 60
Jo(1005)'s score: 60
Park(1000)'s score: 70
Choi(1001)'s score: 70
Ahn(1008)'s score: 70
Kim(1006)'s score: 75
Oh(1002)'s score: 80
Jang(1009)'s score: 85
Son(1004)'s score: 95
Chung(1007)'s score: 100

####################
# Find Test
PASS
PASS
PASS
```

```
####################
# Deletion Test
----------
Node nums: 8
Lee(1010)'s score: 50
Hahn(1003)'s score: 60
Jo(1005)'s score: 60
Kim(1006)'s score: 75
Oh(1002)'s score: 80
Jang(1009)'s score: 85
Son(1004)'s score: 95
Chung(1007)'s score: 100
----------
Node nums: 7
Lee(1010)'s score: 50
Hahn(1003)'s score: 60
Jo(1005)'s score: 60
Kim(1006)'s score: 75
Oh(1002)'s score: 80
Jang(1009)'s score: 85
Chung(1007)'s score: 100
----------
Node nums: 7
Lee(1010)'s score: 50
Hahn(1003)'s score: 60
Jo(1005)'s score: 60
Kim(1006)'s score: 75
Oh(1002)'s score: 80
Jang(1009)'s score: 85
Chung(1007)'s score: 100
----------
Node nums: 5
Lee(1010)'s score: 50
Kim(1006)'s score: 75
Oh(1002)'s score: 80
Jang(1009)'s score: 85
Chung(1007)'s score: 100
```

3. 주어진 조건을 만족하는 HeapADT 와 HeapSort 를 구현하시오. (35 점)

A. 힙(heap)은 다음의 조건을 만족함

i. Min-heap이 되도록 하시오.

ii. Heap에는 학생들의 성적 구조체(STUDENT_GRADE)가 담기며, 구조체의 형태는 다음과 같다.

```
typedef struct
{
    studentId id;              // student ID
    char name[20];             // student Name
    int score;                 // student's score
} STUDENT_GRADE;
```

Heap 의 각 노드(node)에는 학생의 학번(id), 이름(name), 성적(score)이 담겨있다.
자세한 사항은 heapADT.h 파일 참조할 것.

iii. Heap은 성적 구조체에서 성적을 키(key)로 사용한다.

iv. 본 heap은 키의 중복을 허용한다.

B. 다음의 함수들을 구현하시오. (20 점)

- studentInsert (10 점)
  - 학생의 성적을 키(key)로 사용하여 min-heap이 되도록 heap에 학생을 삽입
  - 동일 키 값을 삽입할 때 이동 방향은 마음대로 구현 가능
- studentDelete (10 점)
  - 주어진 heap의 루트(root)에 있는 학생 삭제
  - 중복 키 값이 여러 개 있어도 하나만 삭제

i. 이 외에도 heapADT 에서 "TODO" 영역이 있는 나머지 함수도 모두 구현할 것.

ii. 필요에 따라 함수 추가 가능.

C. Heap 을 이용하여 학생들의 성적을 오름차순으로 정렬하는 heapSort 함수를 구현하시오. (15 점)

i. 성적순으로 정렬된 데이터를 outArr 에 넣어 반환하시오.

출력예시

```
####################
# Insertion Test
PASS
PASS
PASS
PASS
PASS
PASS
PASS
PASS
PASS
PASS
PASS
----------
Num items: 11
Lee(2010)'s score: 50
Hahn(2017)'s score: 60
Jo(2015)'s score: 60
Choi(2019)'s score: 70
Park(2020)'s score: 70
Oh(2018)'s score: 80
Kim(2014)'s score: 75
Chung(2013)'s score: 100
Ahn(2012)'s score: 70
Son(2016)'s score: 95
Jang(2011)'s score: 85
```

```
####################
# Deletion Test
PASS
PASS
PASS
PASS
PASS
PASS
PASS
PASS
PASS
PASS
PASS

####################
# HeapSort Test
PASS
PASS
PASS
PASS
PASS
PASS
PASS
PASS
PASS
PASS
PASS
```

4. Figure 2 에 주어진 그래프(graph)에서 최소 신장 트리(minimum-spanning tree)를 생성하기 위한 Prim 알고리즘을 작성하시오. (25 점)

   A. 다음은 Prim 알고리즘의 수행 방법을 간략히 나타낸 것이다.

A vertex is assumed to be set as the starting vertex. Selecting min. and connected edges in a greedy way.

   B. 주어진 그래프는 행렬(matrix)로 작성되어 있음.

   C. 그래프의 'a' 노드로부터 출발할 것.

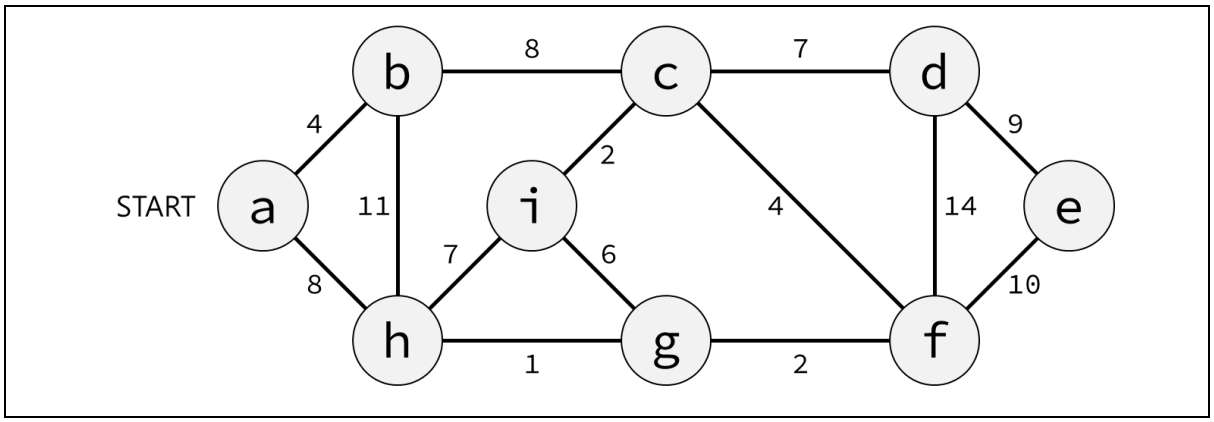
   D. 선택 가능한 노드가 여러 개 있을 경우 빠른 알파벳의 노드를 선택

      i. 'b', 'c' 노드가 동시에 선택 가능할 경우 'b' 노드 선택

   E. 알고리즘의 결과로 exePrim 함수의 distOut 변수에 'a'노드로부터 각 노드까지의 거리를 작성

   F. 출력 예시는 minimum-spanning tree 가 완성되었을 때 'a' 노드로부터 각 노드까지의 거리가 올바로 계산 되었는지 확인함

Figure 2 Graph



출력예시

```
a: PASS
b: PASS
c: PASS
d: PASS
e: PASS
f: PASS
g: PASS
h: PASS
i: PASS
```