2022학년도 2학기 자료구조 중간고사

2022. 10. 18

점수	

학번: **성명:**

※ (1 - 10) 아래 문제를 읽고 물음에 답하시오. [50점 각 5점]

1. 다음 탐색 알고리즘에 대한 설명으로 옳지 않은 것은?

```
def a_search(arr, key) :
  for i in range(len(arr)) :
    if arr[i]==key :
        return i
  return -1
```

① 탐색에 성공하면 키 값의 인덱스를 반환한다.
② 최선의 경우, 시간 복잡도는 빅오 표기법으로 O(1)이다.
③ 최악의 경우, 시간 복잡도는 빅오 표기법으로 O(n)이다.
④ 평균적인 경우, 시간 복잡도는 빅오 표기법으로 O((n + 1)/2)이다.

2. 다음은 1차원 리스트 arr에서 두 번째로 큰 수를 찾는 파이썬 언어로 구현한 함수이다. ㉠, ㉡에 들어갈 내용을 바르게 연결한 것은? (단, 리스트 arr의 모든 원소는 서로 다른 양의 정수이고, 리스트 arr의 크기는 2보다 크거나 같다)

```
def Largest2nd(arr):
    max1=0
    max2=0

    for i in range(len(arr)):
        if (      ㉠      ):
            max2=max1;
            max1=arr[i];
        elif (      ㉡      ):
            max2=arr[i]

    print("second Largest Number = "+str(max2))
```

	㉠	㉡
①	**arr[i] > max1**	**arr[i] > max2 and arr[i] < max1**
②	arr[i] > max1	arr[i] < max2 and arr[i] > max1
③	arr[i] < max1	arr[i] > max2 and arr[i] < max1
④	arr[i] < max1	arr[i] < max2 and arr[i] > max1

3. 이중 연결 리스트(doubly linked list)의 클래스를 나타내는 클래스 node에 이전 노드를 가리키는 포인터(=레퍼런스) llink와 다음 노드를 가리키는 포인터 rlink가 있다고 하자. 포인터 new_node가 가리키는 노드를 포인터 p가 가리키는 노드의 왼쪽에 삽입할 때, 문장의 수행 순서를 바르게 나열한 것은?

```
ㄱ. p.llink.rlink =  p.rlink;
ㄴ. p.rlink.llink =  p.llink;
ㄷ. new_node.rlink =  p;
ㄹ. p.llink.rlink = new_node;
ㅁ. p = new_node;
ㅂ. new_node.llink = p.llink;
ㅅ. p.llink = new_node;
```

① ㄱ → ㄴ
② ㅂ → ㅅ → ㄹ → ㄷ
③ ㄹ → ㄷ → ㅂ → ㅅ
④ ㄹ → ㅂ→ ㄷ → ㅁ

4. 다음 연산식을 후위(postfix) 수식으로 변환했을 때, 3번째 연산자는?

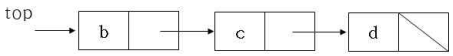
```
1 + (2 - 3 * 4 / (5 + 6)) - 7
```

① +　　② -
③ *　　**④ /**

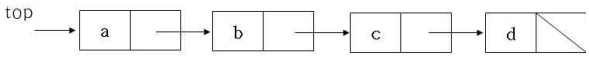
5. 문제 4에서 구한 후위 수식을 스택을 이용하여 계산하고자 한다. 후위 수식의 계산과정에서 스택에 여섯 번째로 삽입(push)되는 값은?

① 3　　② 4
③ 5　　④ 6

6. 다음은 연결 리스트를 이용하여 스택을 표현한 것이다. 이에 대한 설명으로 옳지 않은 것은? (단, push는 스택에 자료를 삽입하는 연산이고, pop은 스택에서 자료를 삭제하는 연산이다)



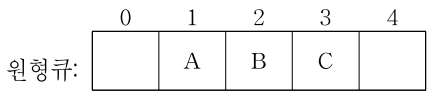
(ㄱ) push 연산 수행 전 스택



(ㄴ) push 연산 수행 후 스택

① 스택에 가장 최근에 입력된 자료는 top이 지시한다.
② 스택에 입력된 자료 중 d가 가장 오래된 자료이다.
③ (ㄴ)에서 자료 c를 가져오려면 pop 연산이 2회 필요하다.
④ (ㄱ)에서 자료의 입력된 순서는 d, c, b이다.

7. 다음 원형 큐(circular queue)에 대하여 연산들(가~바)이 차례대로 수행된 후에 큐의 상태로 옳은 것은? (단, front는 0이고 rear는 3이며, 원형 큐의 empty상태와 full상태는 별도의 카운터를 두어서 처리한다)



가. D를 삽입 나. E를 삽입
다. 두 개의 원소를 삭제 라. F를 삽입
마. 하나의 원소를 삭제 마. G를 삽입

①

0	1	2	3	4
F	G		C	D

②

0	1	2	3	4
F	G			D

③

0	**1**	**2**	**3**	**4**
E	**F**	**G**		**D**

④

0	1	2	3	4
G		B	C	D

8. 다음은 파이썬 언어를 사용하여 원형 큐(circular queue)의 삽입(enqueue)과 삭제(dequeue) 연산을 구현한 것이다.

```
class CircularQueue:
    def __init__(self):
        self.QUEUE_SIZE = 8
        self.Q = []
        for i in range(self.QUEUE_SIZE):
            self.Q.append(0)
        self.front = 0
        self.rear = 0

    def enqueue(self, x):
        self.rear = (self.rear+1)% self.QUEUE_SIZE
        if self.front == self.rear :
            print("Queue is full")
            return
        self.Q[self.rear] = x

    def dequeue(self):
        if self.front == self.rear:
            print("Queue is empty.")
            return
        else :
            self.front = (self.front+1)%self.QUEUE_SIZE
            item = self.Q[self.front]
            self.Q[self.front] = 0
            return item
```

front와 rear의 값이 0인 공백 큐에 6개의 값 1, 3, 2, 4, 8, 6을 차례대로 삽입하고, 여기에서 2개의 값을 삭제한 다음, 다시 3개의 값 5, 7, 9를 차례대로 삽입할 때, 리스트 Q의 최종 모습은?

①

Q[0]	Q[1]	Q[2]	Q[3]	Q[4]	Q[5]	Q[6]	Q[7]
0	7	9	2	4	8	6	5

②

2	4	8	6	5	7	9	0

③

9	0	2	4	8	6	5	7

④

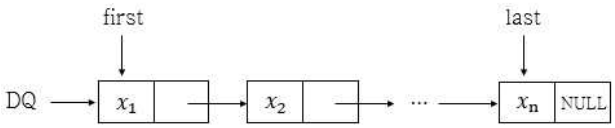
7	**9**	**0**	**2**	**4**	**8**	**6**	**5**

9. 다음 해시구조는 개방 주소법(open addressing) 중 선형 검색법(linear probing)을 사용하여 충돌을 해결하는 예제이다. 해시함수가 '입력되는 색인키의 첫 번째 문자에 대한 알파벳 순위'라고 가정할 때, 버킷 테이블의 ㉠, ㉡에 대한 접근 횟수로 옳게 짝지어진 것은? (예로, 해싱함수 h(alpha)=0, h(beta)=1, h(computer)=2, h(data)=3, h(email)=4, h(father)=5 등을 의미한다)

버킷	색인키	탐사에 필요한 버킷 접근 횟수
0	ascii	1
1	atoi	2
2	char	1
3	define	1
4	equal	1
5	ceil	㉠
6	for	㉡
…		

	㉠	㉡
①	3	1
②	**4**	**2**
③	1	1
④	4	1

10. 데크(deque: double-ended queue)는 삽입과 삭제가 양끝에서 임의로 수행되는 자료구조이다. 다음 그림과 같이 단순 연결리스트(singly linked list)로 데크를 구현한다고 할 때 O(1) 시간 내에 수행할 수 없는 연산은? (단, first와 last는 각각 데크의 첫 번째 원소와 마지막 원소를 가리키며, 연산이 수행된 후에도 데크의 원형이 유지되어야 한다)



① insertFirst 연산: 데크의 첫 번째 원소로 삽입
② insertLast 연산: 데크의 마지막 원소로 삽입
③ deleteFirst 연산: 데크의 첫 번째 원소를 삭제
④ deleteLast 연산: 데크의 마지막 원소를 삭제

11. 인덱스가 0에서 10인 11개의 버켓(bucket)을 갖는 해시 테이블(hash table)이 있다. 해시 함수 $h(k) = k\%\ 11$을 사용하여 키(key) 113, 40, 17, 48, 128, 36, 62를 주어진 순서대로 해시 테이블에 저장하려고 한다. 다음 물음에 답하시오. (단, %는 mod 연산자이고, 각 버켓에는 하나의 키만 저장할 수 있다)

1) 충돌해결을 위해서 선형 탐사(linear probing) 방법을 사용할 때 모든 키를 저장한 후에 해시 테이블을 도시하시오. 그리고 선형 탐사 방법으로 모든 키를 저장한 후 성공적 탐사에서 비교되는 버켓 수의 평균값을 구하시오. (5점)

index	0	1	2	3	4	5	6	7	8	9	10
key				113	48	36	17	40	128	62	

[평균 비교 횟수]

Load factor $\alpha = \frac{N}{M} = \frac{7}{11} = 0.64$

성공적인 탐사에서 평균 비교 횟수

= $\frac{1}{2}\left(1+\frac{1}{1-\alpha}\right) = \frac{1}{2}\left(1+\frac{1}{1-0.64}\right) = 0.38$

(계산은 끝까지 하지 않아도 됨)

2) 충돌해결을 위하여 이차 탐사(quadratic probing) 방법을 사용할 때 모든 키를 저장한 후에 해시 테이블을 도시하시오. 그리고 선형 탐사 방법으로 모든 키를 저장한 후 성공적 탐사에서 비교되는 버켓 수의 평균값을 구하시오. (5점)

index	0	1	2	3	4	5	6	7	8	9	10
key	62	36		113	48		17	40	128		

[평균 비교 횟수]

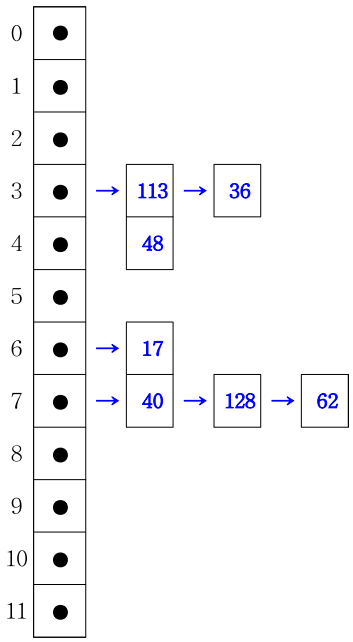
Load factor $\alpha = \frac{N}{M} = \frac{7}{11} = 0.64$

성공적인 탐사에서 평균 비교 횟수

= $1 - \ln(1-\alpha) - \frac{\alpha}{2} = 1 - \ln(1-0.64) - \frac{0.64}{2} = 1.7$

(계산은 끝까지 하지 않아도 됨)

3) 충돌해결을 위하여 체이닝(chaining) 기법을 사용할 때 모든 키를 저장한 후에 해시 테이블을 도시하시오. (5점)



4) 삽입 및 탐색 효율의 관점에서 체이닝 기법이 선형탐사법에 비해 갖는 장점을 서술하고, 1)과 3)의 결과로 부터 구체적인 사례를 들어 설명하시오. (5점)

선형탐사는 α값이 1에 근접할수록 군집화가 심화되어 탐사 실패로 인한 평균 비교 횟수가 증가한다. 예를 들어 key값 62를 탐사하는데 선형탐사는 3번의 비교만에 탐사에 성공하지만, 체이닝은 1번의 비교만으로 탐사에 성공한다.

* 각 주소방식 별로 주어진 key 값을 찾는 데 필요한 평균 비교(comparison) (또는 탐사(probe)) 횟수를 α에 관해 정리하면 아래와 같다.

	선형탐사	이차탐사	이중해싱	체이닝
탐사 성공	$\frac{1}{2}\left(1+\frac{1}{1-\alpha}\right)$	$1-\ln(1-\alpha)-\frac{\alpha}{2}$	$\frac{1}{\alpha}\ln(\frac{\alpha}{1-\alpha})$	$1+\frac{\alpha}{2}$
탐사 실패	$\frac{1}{2}\left(1+\frac{1}{(1-\alpha)^2}\right)$	$\frac{1}{1-\alpha}-\alpha-\ln(1-\alpha)$	$\frac{1}{1-\alpha}$	α