ĐỀ KIỂM TRA CUỐI KỲ (2/2018-2019)

MÔN THI: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

THỜI GIAN: 120 PHÚT

NGÀY THI: 30/05/2019

HƯỚNG DẪN:

- Sinh viên điền Họ tên và MSSV vào đề thi. Sinh viên viết trực tiếp câu trả lời lên đề thi và nộp lại đề thi.
- Được sử dụng tài liệu trên giấy.
- Không sử dụng laptop, điện thoại, hoặc các thiết bị tương tự
- Tổng điểm: 110

Họ và tên: _____

MSSV: _____

PHẦN	ĐIỂM
Phần A (15 điểm)	
Phần B (15 điểm)	
Phần C (20 điểm)	
Phần D (20 điểm)	
Phần E (15 điểm)	
Phần F (10 điểm)	
Phần G (15 điểm)	
TỔNG ĐIỂM	

Ngày ra đề: 28/05/2019

TRƯỞNG KHOA/BỘ MÔN **GIẢNG VIÊN**

TS. Nguyễn Hồ Mẫn Rạng

PHẦN A – PHÂN TÍCH ĐỘ PHỨC TẠP (15 điểm)

A1 (**L.O.1.1**). Điền vào các ô trống độ phức tạp trong trường hợp trung bình (average case) của các hàm thêm, xóa, và tìm kiếm phần tử trong các cấu trúc dữ liệu sau (**5 điểm**):

Trả lời:

	Mảng được sắp xếp (Sorted ArrayList)	Cây AVL (AVL Tree)	Bảng Hash (Hash Table)
Thêm (Insertion)			
Xóa (Deletion)			
Tìm kiếm (Retrieval)			

A2(**L.O.1.1**). Giả sử bạn thu thập được thông tin dữ liệu thời gian chạy của một chương trình theo kích thước số lượng phần tử N của dữ liệu đầu vào như sau:

N	Time
1,000	1.00 s
8,000	32.00 s
64,000	1,024.00 s
512,000	32,768.00 s

Ước lượng thời gian chạy f(N) theo kích thước N. (**5 điểm**)

Trả lời:

A3(**L.O.1.1**). Cho biết độ phức tạp của thuật toán bên dưới (**5 điểm**)

```
for (int i = 0; i < n; i++) {
        for (int j = i; j < n - i; j++) {
                count++;
        }
}
```

Trả lời:

PHẦN B – DANH SÁCH (20 điểm)

Sử dụng cấu trúc dữ liệu danh sách liên kết đôi như bên dưới để trả lời các câu B1-B2

```
class Node {
public:
	int data;
	Node *next,*prev;
};
```

```
class DoubleLinkedList {
	Node *head, *tail;
	int count;

};
```

B1(**L.O.2.1**). Biết rằng kiểu `int` có kích thước là 4 bytes và kiểu con trỏ (pointer) có kích thước là 8 bytes. Hỏi kích thước của một đối tượng kiểu `DoubleLinkedList` chứa 10 phần tử là bao nhiêu bytes? (**5 điểm**)

Trả lời

B2(**L.O.2.2**). Cho biết độ phức tạp trong trường hợp xấu nhất (worst-case) đối với những phương thức sau của lớp `DoubleLinkedList`. (**5 điểm**)

Trả lời

Phương thức	Mô tả	Độ phức tạp
`addFirst(item)`	Thêm một phần tử **`item`** vào đầu danh sách	
`get(idx)`	Trả về phần tử thứ **`idx`** trong danh sách	
`set(idx, item)`	Thay thế phần tử thứ **`idx`** trong danh sách bằng **`item`**	
`removeLast()`	Xóa phần tử cuối cùng trong danh sách	
`contains(item)`	Tìm xem phần tử **`item`** có xuất hiện trong danh sách không?	

B3(**L.O.3.1**). Giả sử bạn cần hiện thực chức năng quản lý các ứng dụng đang được thực thi trên hệ điều hành. Thông thường, hệ điều hành sẽ đặt các ứng dụng đang thực thi trên một danh sách, sau đó lần lượt duyệt qua chúng, và yêu cầu CPU thực thi mỗi ứng dụng trong một khoảng thời gian nhất định. Nếu ứng dụng đó chưa hoàn tất thì sẽ được đặt trở lại danh sách và chờ trong khi CPU thực thi các ứng dụng khác. Hãy lựa chọn cấu trúc dữ liệu để thực hiện chức năng này hiệu quả nhất. (**5 điểm**)

Gợi ý: Khoanh tròn MỘT lựa chọn duy nhất cho mỗi hàng

Trả lời:

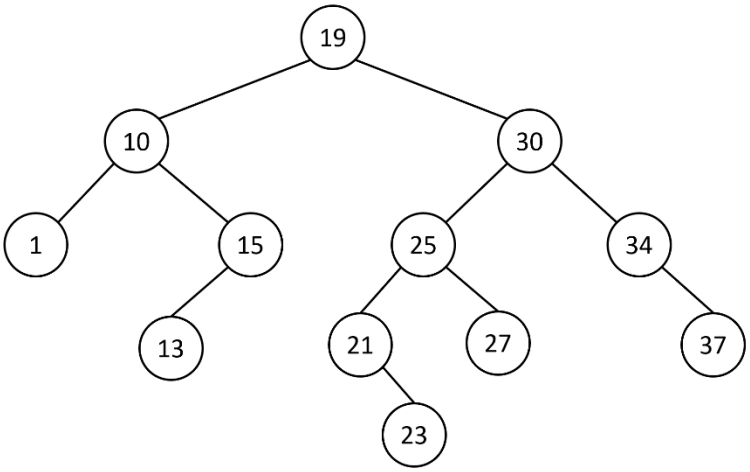
Loại:	**a. ArrayList**	**b. LinkedList**	**c. Stack**	**d. Queue**
Hiện thực dựa trên:	**a. Mảng**	**b. Liên kết đơn**	**c. Liên kết đôi**	
Hiện thực vòng:	**a. Có**	**b. Không**		

PHẦN C – CÂY NHỊ PHÂN (20 điểm)

C1(**L.O.2.1**)Vẽ cây nhị phân tìm kiếm (BST) sau khi thực hiện thêm lần lượt các phần tử sau đây: 1, 5, 3, 4, 2, 8, 9. (**5 điểm**)

Trả lời:

C2(**L.O.2.2**). Cho cây AVL như bên dưới. Vẽ lại cây AVL sau khi xóa node chứa giá trị **1** (**5 điểm**)



Trả lời:

C3(**L.O.2.3**). Hiện thực phương thức `Node* maxNode()` cho cấu trúc dữ liệu `BST` được mô tả như bên dưới. Phương thức này sẽ trả về địa chỉ NULL nếu cây rỗng, ngược lại sẽ trả về địa chỉ của node chứa giá trị lớn nhất trong cây nhị phân tìm kiếm. (**10 điểm**)

```
struct Node {
	int val;
	Node * left, *right;
};
```

```
class BST
{
	Node * root;
};
```

Lưu ý: điểm sẽ được đánh giá dựa vào độ chính xác và hiệu quả của giải thuật.

Trả lời:

```
Node* BST::maxNode() {
	// YOUR CODE HERE









}
```

PHẦN D – MAX HEAP (20 điểm)

D1(**L.O.2.1**). Cho max-heap được lưu trữ dưới dạng mảng chứa các phần tử sau: 8, 5, 7, 2, 4, 6, 1. Cho biết thứ tự lưu trữ của các phần tử sau khi thêm phần tử 9 vào max-heap này. (**5 điểm**)

Trả lời:

Index	0	1	2	3	4	5	6	7
Value								

D2(**L.O.2.1**). Cho max-heap được lưu trữ dưới dạng mảng chứa các phần tử sau: 10, 5, 7, 2, 3, 6, 1. Cho biết thứ tự lưu trữ của các phần tử còn lại sau khi xóa 10 ra khỏi max-heap này. (**5 điểm**)

Trả lời:

Index	0	1	2	3	4	5	6	7
Value								

D3(**L.O.2.3**). Hiện thực phương thức `void replace(int idx, int val)` trong class MaxHeap. Phương thức này sẽ thay thế giá trị tại vị trí `idx` trong max-heap bằng giá trị `val` và thực hiện các phép điều chỉnh cần thiết (`reHeapUp`, hoặc `reHeapDown`) để vẫn thỏa mãn điều kiện của max-heap. Giả sử rằng chỉ số `idx` luôn hợp lệ và các phương thức `reHeapUp` () và `reHeapDown` () đã được hiện thực. (**10 điểm**)

Ví dụ: Cho max-heap: 8, 5, 7, 2, 4, 6, 1.

a. Nếu gọi phương thức replace(1, 9), thì max-heap sẽ được điều chỉnh thành: 9, 8, 7, 2, 4, 6, 1
b. Nếu gọi phương thức replace(1, 3), thì max-heap sẽ được điều chỉnh thành: 8, 4, 7, 2, 3, 6, 1

Trả lời:

```
class MaxHeap {
	int capacity, size;
	int* a;
	void reHeapUp(int pos);
	void reHeapDown(int pos);
public:
	…
	void replace(int idx, int val);
};

void MaxHeap::replace(int idx, int val) {
	// YOUR CODE HERE









}
```

PHẦN E – SẮP XẾP (15 điểm)

E1(**L.O.4.1**). Giả sử bạn đang sử dụng giải thuật Quicksort để sắp xếp một mảng các phần tử số nguyên. Bên dưới là kết quả sau lượt phân chia đầu tiên (partition). Hỏi những phần tử nào có thể là phần tử pivot? (**3 điểm**)

1, 4, 3, 6, 5, 7, 10, 8, 20, 19, 25

Trả lời:

E2(**L.O.4.1**). Cho biết độ phức tạp của các giải thuật sắp xếp sau đối với dữ liệu đầu vào là các mảng đã được sắp xếp theo đúng thứ tự và các mảng đã được sắp xếp theo thứ tự ngược lại. Hãy điền vào các ô trống độ phức tạp của các giải thuật: (**7 điểm**)

Gợi ý: Chọn một trong các độ phức tạp sau để điền vào các ô trống: O(1), O(N), O(N log N) , O (N^2)

Trả lời

	Đã được sắp xếp theo đúng thứ tự (sorted order)	Đã được sắp xếp nhưng theo thứ tự ngược lại (reverse order)
Heapsort		
Mergesort		
Insertion sort		
Selection sort		

E3(**L.O.4.2**). Hãy mô tả ngắn gọn **TỐI THIỂU 3** giải pháp để cải thiện độ hiệu quả của giải thuật Quicksort mô tả như bên dưới (**5 điểm**)

```
int divide(int* a, int lo, int hi)
{
	int pivot = a[hi];
	int i = lo;
	for (int j = lo; j < hi; j++) {
		if (a[j] < pivot) {
			swap(a[j], a[i]);
			i++;
		}
	}
	swap(a[i], a[hi]);
	return i;
}
```

```
void quickSort(int* a, int lo, int hi)
{
	if (low < high) {
		int p = divide(a, lo, hi);
		quickSort(a, lo, p - 1);
		quickSort(a, p + 1, hi);
	}
}
```

Trả lời:

PHẦN F – BẢNG HASH (10 điểm)

Cho bảng hash có kích thước là m = 11, hàm hash f(x) = x%m, và sử dụng kỹ thuật linear probing để giải quyết đụng độ (collision). Ban đầu bảng hash rỗng (không chứa phần tử nào). Tiến hành lần lượt thêm các phần tử sau vào bảng hash: 12, 23, 13, 21, 32, 44, 45.

F1(**L.O.4.1**). Cho biết nội dung của bảng hash sau khi thêm các phần tử trên (**5 điểm**)

Trả lời:

Index	0	1	2	3	4	5	6	7	8	9	10
Key											

F2(**L.O.4.1**). Cho biết nội dung của bảng hash sau khi xóa phần tử 12 ra khỏi bảng hash (**5 điểm**)

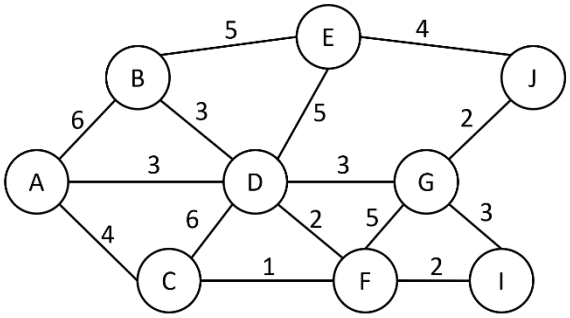
Trả lời:

Index	0	1	2	3	4	5	6	7	8	9	10
Key											

PHẦN G – ĐỒ THỊ (15 điểm)

G1(**L.O.2.1**). Cho đồ thị vô hướng có trọng số như hình bên. Tìm cây phủ tối thiểu của đồ thị trên (Minimum Spanning Tree) **(5 điểm)**

Trả lời:



G2(**L.O.2.1**). Cho biết giải thuật duyệt theo chiều sâu (DFS) như bên dưới (bên trái) và thứ tự các hàm được gọi khi chạy giải thuật duyệt theo chiều sâu trên một đồ thị có hướng (bên phải). Hãy vẽ lại đồ thị có hướng này. (**10 điểm**)

```
void dfs(int v) {
	marked[v] = true;
	cout << v << " ";
	for (int w : adj(v)) {
		if (!isMark(w))
			dfs(w);
	}
}

bool isMark(int v) {
	return marked[v];
}
```

```
dfs(1)
→	isMark(3)
→	dfs(3)
		→	isMark(2)
		→	dfs(2)
				→	isMark(4)
				→	dfs(4)
				→	isMark(5)
				→	dfs(5)
		→	isMark(4)
		→	isMark(5)
→	isMark(2)
```

Trả lời

- HẾT -