TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH

Họ và tên:.....
MSSV:.....

KIỂM TRA CUỐI KỲ (25/07/2020)

Môn thi: Cấu trúc dữ liệu và giải thuật Thời gian làm bài: 120 phút

☐ Không được phép dùng tài liệu ☒ Được phép dùng tài liệu giấy
Đề thi số: **1** **Đề thi gồm 6 trang.**

Chú ý: Sinh viên làm trực tiếp trên đề thi; mỗi câu hỏi trắc nghiệm chỉ chọn một đáp án đúng nhất. Các khai báo về thư viện, các hàm cơ bản và khai báo khác (nếu cần thiết) được xem như đầy đủ

I. TRẮC NGHIỆM (3 điểm)

Câu 1. Cho array A chứa n số nguyên có giá trị khác nhau từng đôi một. Giả sử muốn tìm giá trị nhỏ nhất trong A, trong trường hợp xấu nhất, giải pháp nào sau đây hiệu quả nhất:

(A) Sort A tăng dần, min là phần tử đầu tiên
(B) Lặp qua cả dãy tìm phần tử min
(C) Tạo 1 BST từ A, tìm phần tử min
(D) Tạo 1 MinHeap từ A, min được chứa trong root

Câu 2. Chọn phát biểu sai

(A) Linear Searching có độ phức tạp O(n)
(B) Binary Searching trên Ordered Array có độ phức tạp O(log n)
(C) Searching trên cây AVL có độ phức tạp O(ln n)
(D) Searching trên Heap có độ phức tạp O(log n)

Câu 3. Giả sử cần chuyển biểu thức trung tố sang hậu tố, cấu trúc dữ liệu nào phù hợp nhất được sử dụng để giải quyết bài toán này:

(A) Queue (B) List
(C) Tree (D) Stack

Câu 4. Trong trường hợp tốt nhất, giải thuật sort nào kém hiệu quả nhất:

(A) Insertion sort (B) Quick sort
(C) Bubble sort (D) Selection sort

Câu 5. Xét đoạn code sau, độ phức tạp Big-O:

```
int test = 0, i, j;
for (i = 0; i < n; i++):
    for (j = 0; j < i; j++):
        test = test + i * j;
```

(A) O(n) (B) O(n^2)
(C) O(log n) (D) O(1)

Câu 6. Dạng nào của List phù hợp để trả lời câu hỏi "Phần tử thứ i của list chứa gì?":

(A) Lists hiện thực bằng 1 array
(B) Singly-linked lists
(C) Doubly-linked lists (danh sách liên kết 2 chiều)
(D) Singly-linked lists hoặc Doubly-linked lists đều phù hợp

Câu 7. Thuật ngữ nào dùng để diễn tả giải thuật O(n):

(A) Constant (B) Linear
(C) Logarithmic (D) Cubic

Câu 8. Trong trường hợp tốt nhất, giải thuật sort nào hiệu quả nhất:

(A) Insertion sort (B) Quick sort
(C) Merge sort (D) Selection sort

Câu 9. Số node tối thiểu của 1 **full** binary tree có depth = 3 (cây chỉ có root xem như có depth = 0) là:

(A) 7 (B) 3
(C) 5 (D) 15
(E) 8

Câu 10. Con trỏ (pointers gồm front, rear) nào sẽ bị thay đổi khi insert 1 phần tử vào 1 queue đang có 1 phần tử:

(A) front (B) rear
(C) cả 2
(D) không có con trỏ nào thay đổi

Câu 11. 1 cây nhị phân khi duyệt NLR cho kết quả **14 2 1 3 11 10 7 30 40**, duyệt LRN cho kết quả **1 3 2 7 10 40 30 11 14**, duyệt LNR sẽ có kết quả là:

(A) 1 2 3 7 10 11 14 30 40 (B) 1 2 3 14 7 10 11 40 30
(C) 1 2 3 7 10 14 30 11 40 (D) 14 1 2 3 10 11 7 40 30

Câu 12. 1 hash table có length 10 sử dụng hash function h(k) = k mod 10, và linear probling. Sau khi chèn 6 giá trị vào 1 empty hash table, thì table chứa giá trị như sau: [_, _, 42, 23, 34, 52, 46, 33, _, _] (index của table được đánh từ 0). Các key values được chèn theo thứ tự trước sau như thế nào?

(A) 46, 42, 34, 52, 23, 33 (B) 34, 42, 23, 52, 33, 46
(C) 46, 34, 42, 23, 52, 33 (D) 42, 46, 33, 23, 34, 52

Câu 13. Xét giải thuật dùng stack để xác định sự cân bằng của 1 chuỗi các dấu đóng, mở ngoặc. Số phần tử trong stack tại một thời điểm (AT ANY ONE TIME) lớn nhất là bao nhiêu khi phân tích chuỗi sau: (()(())(()))

(A) 5 hoặc nhiều hơn (B) 1
(C) 2 (D) 3
(E) 4

II. ĐIỀN KẾT QUẢ (4 điểm)

Câu 14. Cho function sau:

```
int mystery_2(Node *tree = this.root):
    if(tree == nullptr)
        return 0;
    else if (tree.getLeft() == nullptr and tree.getRight() == nullptr):
        return 1;
    return 1 + mystery_2(tree.getLeft()) + mystery_2(tree.getRight());
```

Hãy cho biết công dụng của function này:

Đếm số node có trong cây

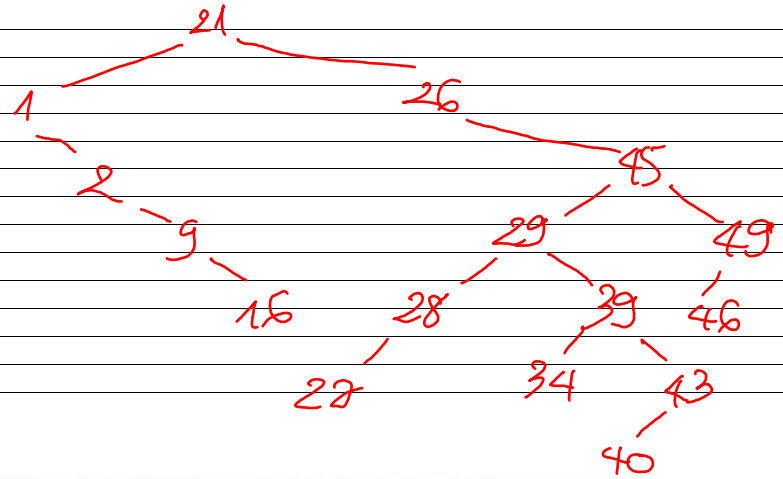
Câu 15. Cho function sau:

```
int mystery_1(Node *tree = this.root):
    if(tree == nullptr):
        return 0;
    elif (tree.getLeft() == nullptr and tree.getRight() == nullptr):
        return 0;
    return 1 + mystery_1(tree.getLeft()) + mystery_1(tree.getRight());
```

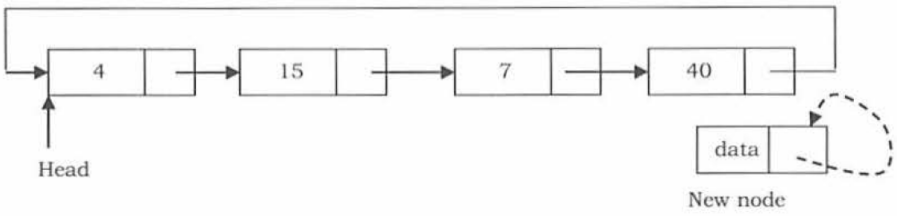
Hãy cho biết công dụng của function này:

Đếm số node có con trong cây

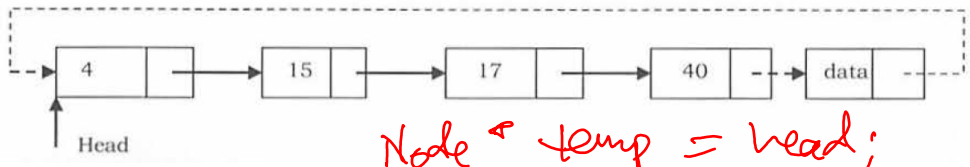
Câu 16. Cho array A = [21, 1, 26, 45, 29, 28, 2, 9, 16, 49, 39, 27, 43, 34, 46, 40], lần lượt insert các phần tử của A theo thứ tự index từ 0->15 vào 1 empty BST, hãy vẽ BST cuối cùng được tạo ra.



Câu 17. Cho hình bên dưới:



Hãy viết các câu lệnh cần thiết để có thể chèn phần tử New node vào vị trí như hình sau, giả sử đã có 2 con trỏ Head và New node như hình:




```
Node * temp = head;
while (temp -> next != head)
    temp = temp -> next;
New_node -> next = head;
temp -> next = New_node;
```

b1 : 17 54 26 93 20 77 31 44 55

b2 : 20 54 26 93 31 77 44 55

b3 : 26 54 31 93 44 77 55

b4 : 31 54 44 93 55 77

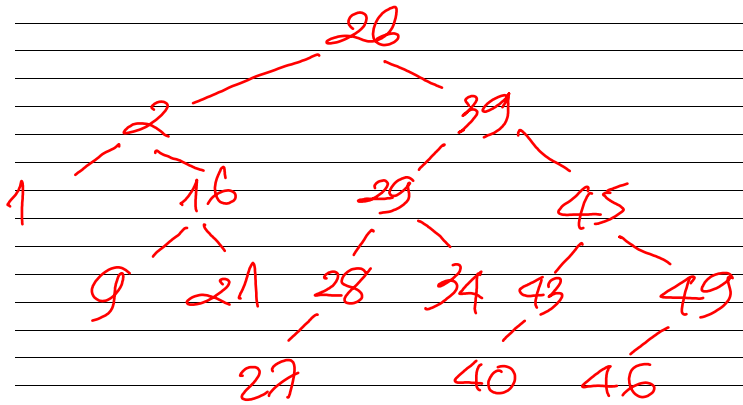
Câu 18. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], sử dụng **bubble sort** sắp xếp A **tăng dần**, sau 4 bước sắp xếp (four complete passes) thì A = ?.

A = [17, 20, 26, 31, 54, 44, 93, 55, 77]

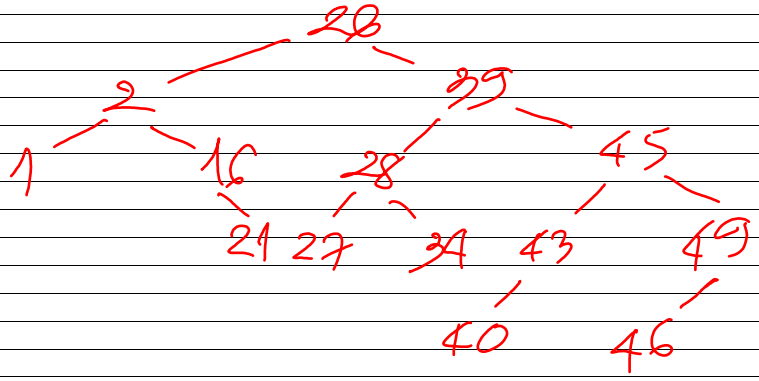
Câu 19. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], sử dụng **insertion sort** sắp xếp A **giảm dần**, sau 4 bước sắp xếp (four complete passes) thì A = ?.

A = [93, 54, 26, 17, 77, 31, 44, 55, 20]

Câu 20. Cho array A = [21, 1, 26, 45, 29, 28, 2, 9, 16, 49, 39, 27, 43, 34, 46, 40], lần lượt insert các phần tử của A theo thứ tự index từ 0->15 vào 1 cây empty AVL, hãy vẽ cây AVL cuối cùng được tạo ra.

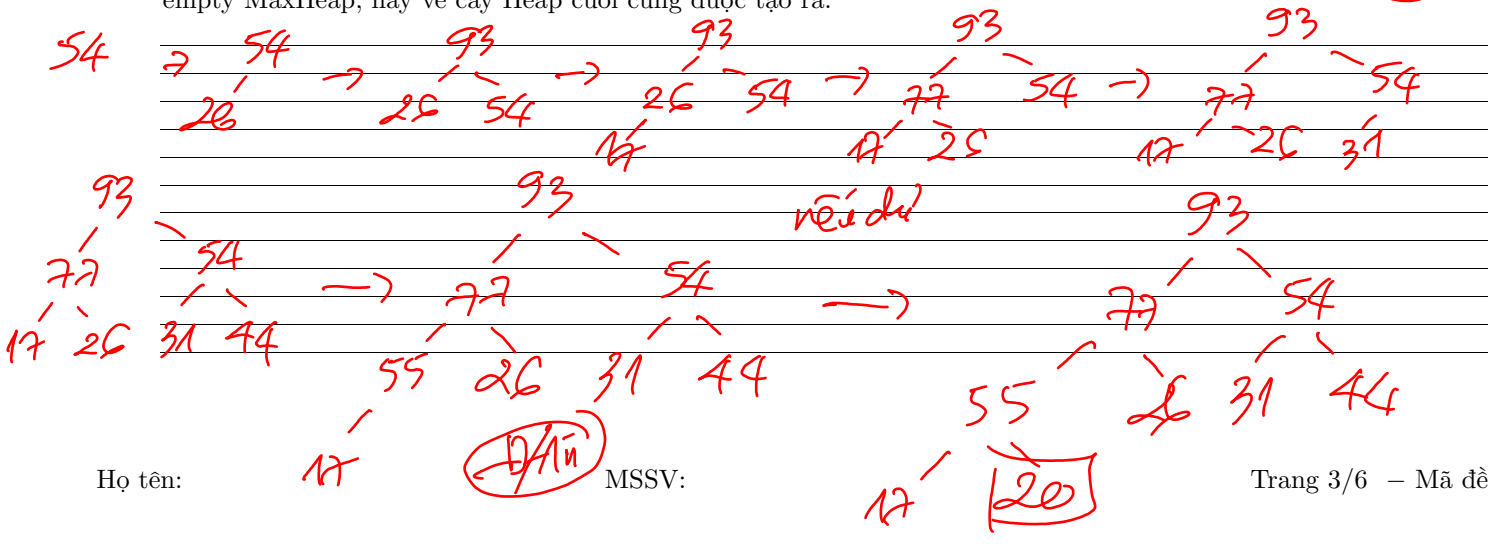


Câu 21. Xóa lần lượt 9, và 29 của cây AVL ở câu trên. Hãy vẽ cây AVL sau khi xóa.

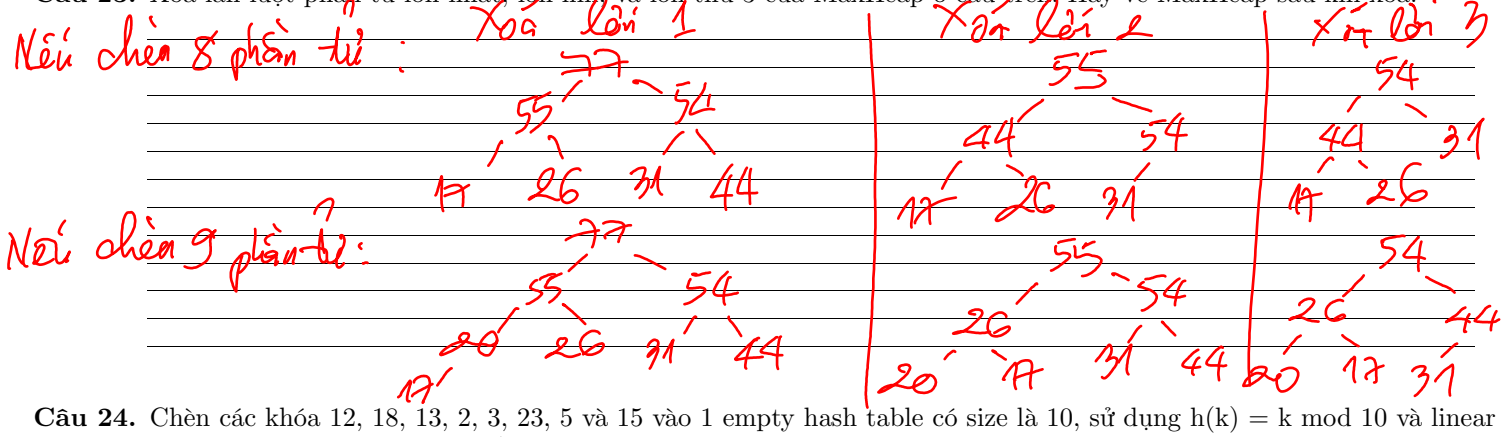


Câu 22. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], lần lượt insert các phần tử của A theo thứ tự index từ 0->7 vào 1 empty MaxHeap, hãy vẽ cây Heap cuối cùng được tạo ra.

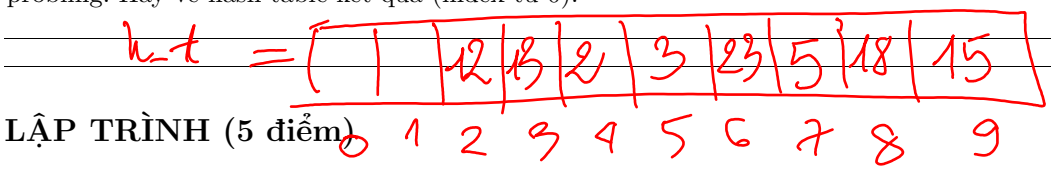
8



Câu 23. Xóa lần lượt phần tử lớn nhất, lớn nhì và lớn thứ 3 của MaxHeap ở câu trên. Hãy vẽ MaxHeap sau khi xóa.



Câu 24. Chèn các khóa 12, 18, 13, 2, 3, 23, 5 và 15 vào 1 empty hash table có size là 10, sử dụng h(k) = k mod 10 và linear probling. Hãy vẽ hash table kết quả (index từ 0).



III. LẬP TRÌNH (5 điểm)

Câu 25. (1đ) (CODING STYLE)
Sinh viên sẽ có điểm câu này nếu làm ít nhất 1 câu trong các câu tiếp theo và có coding style tốt. Điểm được chấm theo 3 mức: 0, 0.5, hoặc 1.

Câu 26. Viết function **bool moveK2first(Node* head, int k)** để di chuyển node ở vị trí thứ k ra đầu danh sách. Giả sử phần tử đầu tiên nằm ở vị trí 1.

Câu 27. Viết function **int countLeave(TreeNode* root)** để đếm số lá có trong 1 cây nhị phân tổng quát.

Câu 28. Cho các khai báo sau:

```
struct StudentNode {
    int id;
    string name;
    StudentNode* next;
};

struct ClassNode {
    int id;
    string name;
    StudentNode* students;
    ClassNode* next;
};
```

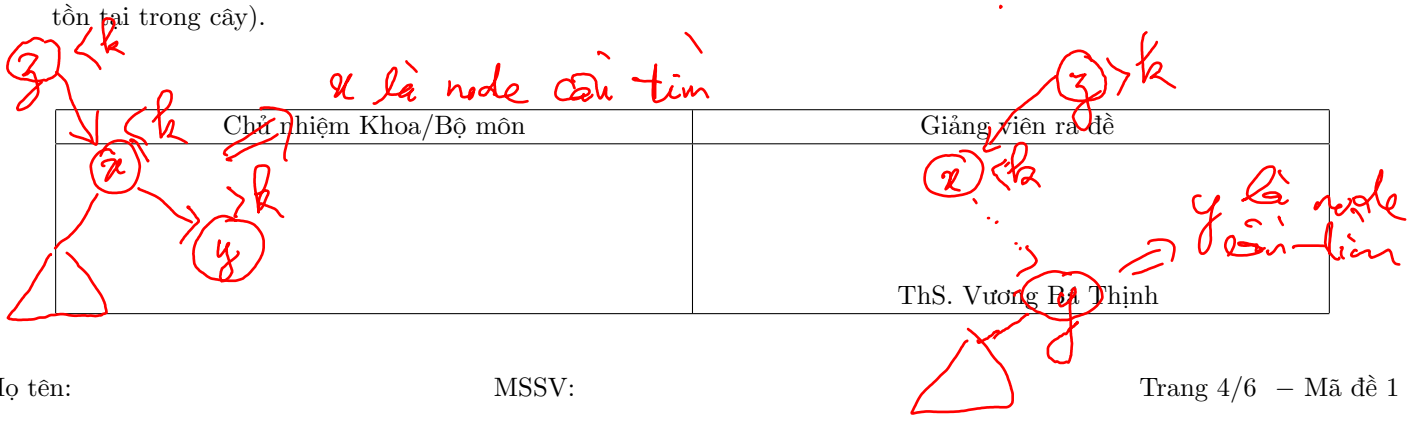
Trong đó, cấu trúc StudentNode dùng để lưu trữ một node trong danh sách liên kết đơn chứa thông tin sinh viên của một lớp, ClassNode dùng để lưu trữ một node trong danh sách liên kết đơn chứa thông tin của các lớp đang quản lý trong một trường học. Thành phần students của ClassNode đại diện cho danh sách sinh viên của lớp đó.

Hãy định nghĩa hàm **bool addStudent(ClassNode* list, int classId, int studentId, string studentName)**, trong đó list là con trỏ trỏ đến phần tử đầu tiên trong danh sách lớp, classId, studentId, studentName lần lượt là mã số lớp, mã số sinh viên và tên của sinh viên. Hàm thực thực hiện chức năng sau:

- Thêm sinh viên vào đúng lớp có classId tương ứng và trả về true. Nếu không tồn tại classId đó, không thực hiện và trả về false.

- Nếu tồn tại studentId (trong toàn bộ các lớp) không thực hiện và trả về false.

Câu 29. Viết function **void printSmallerNearestNode(BSTNode* root, int key)** để tìm kiếm và in ra màn hình node duy nhất có giá trị nhỏ (hoặc bằng) hơn và gần nhất với giá trị **key** trong 1 BST (giá trị **key** vì thế có thể không tồn tại trong cây).



Chủ nhiệm Khoa/Bộ môn	Giảng viên ra đề
	ThS. Vương Bá Thịnh

TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH

KIỂM TRA CUỐI KỲ (25/07/2020)

Môn thi: Cấu trúc dữ liệu và giải thuật Thời gian làm bài: 120 phút

☐ Không được phép dùng tài liệu ☒ Được phép dùng tài liệu giấy
Đề thi số: 1

I. TRẮC NGHIỆM (3 điểm)

Câu 1. Cho array A chứa n số nguyên có giá trị khác nhau từng đôi một. Giả sử muốn tìm giá trị nhỏ nhất trong A, trong trường hợp xấu nhất, giải pháp nào sau đây hiệu quả nhất:

(A) Sort A tăng dần, min là phần tử đầu tiên
[B] Lặp qua cả dãy tìm phần tử min
(C) Tạo 1 BST từ A, tìm phần tử min
(D) Tạo 1 MinHeap từ A, min được chứa trong root

Câu 2. Chọn phát biểu sai

(A) Linear Searching có độ phức tạp O(n)
(B) Binary Searching trên Ordered Array có độ phức tạp O(log n)
(C) Searching trên cây AVL có độ phức tạp O(ln n)
[D] Searching trên Heap có độ phức tạp O(log n)

Câu 3. Giả sử cần chuyển biểu thức trung tố sang hậu tố, cấu trúc dữ liệu nào phù hợp nhất được sử dụng để giải quyết bài toán này:

(A) Queue (B) List
(C) Tree [D] Stack

Câu 4. Trong trường hợp tốt nhất, giải thuật sort nào kém hiệu quả nhất:

(A) Insertion sort (B) Quick sort
[C] Bubble sort (D) Selection sort

Câu 5. Xét đoạn code sau, độ phức tạp Big-O:

```
int test = 0, i, j;
for (i = 0; i < n; i++):
    for (j = 0; j < i; j++):
        test = test + i * j;
```

(A) O(n) [B] O(n^2)
(C) O(log n) (D) O(1)

Câu 6. Dạng nào của List phù hợp để trả lời câu hỏi "Phần tử thứ i của list chứa gì?":

[A] Lists hiện thực bằng 1 array
(B) Singly-linked lists
(C) Doubly-linked lists (danh sách liên kết 2 chiều)
(D) Singly-linked lists hoặc Doubly-linked lists đều phù hợp

Câu 7. Thuật ngữ nào dùng để diễn tả giải thuật O(n):

(A) Constant [B] Linear
(C) Logarithmic (D) Cubic

Câu 8. Trong trường hợp tốt nhất, giải thuật sort nào hiệu quả nhất:

[A] Insertion sort (B) Quick sort
(C) Merge sort (D) Selection sort

Câu 9. Số node tối thiểu của 1 **full** binary tree có depth = 3 (cây chỉ có root xem như có depth = 0) là:

[A] 7 (B) 3
(C) 5 (D) 15
(E) 8

Câu 10. Con trỏ (pointers gồm front, rear) nào sẽ bị thay đổi khi insert 1 phần tử vào 1 queue đang có 1 phần tử:

(A) front [B] rear
(C) cả 2
(D) không có con trỏ nào thay đổi

Câu 11. 1 cây nhị phân khi duyệt NLR cho kết quả **14 2 1 3 11 10 7 30 40**, duyệt LRN cho kết quả **1 3 2 7 10 40 30 11 14**, duyệt LNR sẽ có kết quả là:

(A) 1 2 3 7 10 11 14 30 40
[B] 1 2 3 14 7 10 11 40 30
(C) 1 2 3 7 10 14 30 11 40
(D) 14 1 2 3 10 11 7 40 30

Câu 12. 1 hash table có length 10 sử dụng hash function h(k) = k mod 10, và linear probling. Sau khi chèn 6 giá trị vào 1 empty hash table, thì table chứa giá trị như sau: [_, _, 42, 23, 34, 52, 46, 33, _, _] (index của table được đánh từ 0). Các key values được chèn theo thứ tự trước sau như thế nào?

(A) 46, 42, 34, 52, 23, 33 (B) 34, 42, 23, 52, 33, 46
[C] 46, 34, 42, 23, 52, 33 (D) 42, 46, 33, 23, 34, 52

Câu 13. Xét giải thuật dùng stack để xác định sự cân bằng của 1 chuỗi các dấu đóng, mở ngoặc. Số phần tử trong stack tại một thời điểm (AT ANY ONE TIME) lớn nhất là bao nhiêu khi phân tích chuỗi sau: (()(())(()))

[A] 5 hoặc nhiều hơn (B) 1
(C) 2 (D) 3
(E) 4

II. ĐIỀN KẾT QUẢ (4 điểm)

Câu 14. Cho function sau:

```
int mystery_2(Node *tree = this.root):
    if(tree == nullptr)
        return 0;
    else if (tree.getLeft() == nullptr and tree.getRight() == nullptr):
        return 1;
    return 1 + mystery_2(tree.getLeft()) + mystery_2(tree.getRight());
```

Hãy cho biết công dụng của function này:

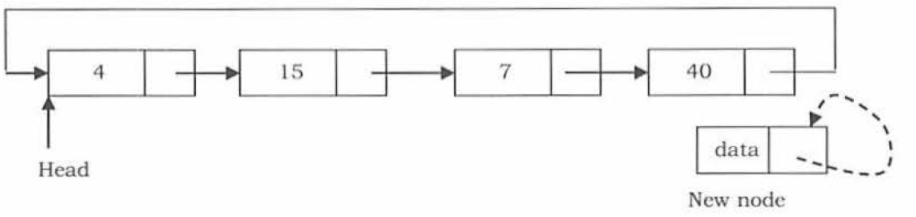
Câu 15. Cho function sau:

```
int mystery_1(Node *tree = this.root):
    if(tree == nullptr):
        return 0;
    elif (tree.getLeft() == nullptr and tree.getRight() == nullptr):
        return 0;
    return 1 + mystery_1(tree.getLeft()) + mystery_1(tree.getRight());
```

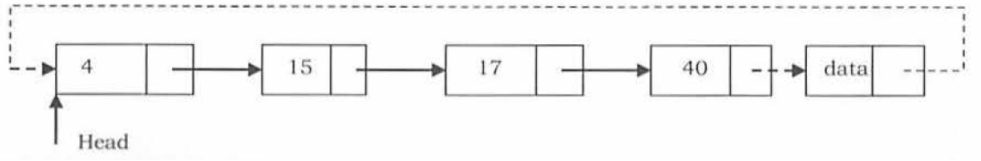
Hãy cho biết công dụng của function này:

Câu 16. Cho array A = [21, 1, 26, 45, 29, 28, 2, 9, 16, 49, 39, 27, 43, 34, 46, 40], lần lượt insert các phần tử của A theo thứ tự index từ 0->15 vào 1 empty BST, hãy vẽ BST cuối cùng được tạo ra.

Câu 17. Cho hình bên dưới:



Hãy viết các câu lệnh cần thiết để có thể chèn phần tử New node vào vị trí như hình sau, giả sử đã có 2 con trỏ Head và New node như hình:



Câu 18. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], sử dụng **bubble sort** sắp xếp A **tăng dần**, sau 4 bước sắp xếp (four complete passes) thì A = ?.

Câu 19. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], sử dụng **insertion sort** sắp xếp A **giảm dần**, sau 4 bước sắp xếp (four complete passes) thì A = ?.

Câu 20. Cho array A = [21, 1, 26, 45, 29, 28, 2, 9, 16, 49, 39, 27, 43, 34, 46, 40], lần lượt insert các phần tử của A theo thứ tự index từ 0->15 vào 1 cây empty AVL, hãy vẽ cây AVL cuối cùng được tạo ra.

Câu 21. Xóa lần lượt 9, và 29 của cây AVL ở câu trên. Hãy vẽ cây AVL sau khi xóa.

Câu 22. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], lần lượt insert các phần tử của A theo thứ tự index từ 0->7 vào 1 empty MaxHeap, hãy vẽ cây Heap cuối cùng được tạo ra.

Câu 23. Xóa lần lượt phần tử lớn nhất, lớn nhì, và lớn thứ 3 của MaxHeap ở câu trên. Hãy vẽ MaxHeap sau khi xóa.

Câu 24. Chèn các khóa 12, 18, 13, 2, 3, 23, 5 và 15 vào 1 empty hash table có size là 10, sử dụng h(k) = k mod 10 và linear probling. Hãy vẽ hash table kết quả (index từ 0).

III. LẬP TRÌNH (5 điểm)

Câu 25. (1đ) (CODING STYLE)
Sinh viên sẽ có điểm câu này nếu làm ít nhất 1 câu trong các câu tiếp theo và có coding style tốt. Điểm được chấm theo 3 mức: 0, 0.5, hoặc 1.

Câu 26. Viết function **bool moveK2first(Node* head, int k)** để di chuyển node ở vị trí thứ k ra đầu danh sách. Giả sử phần tử đầu tiên nằm ở vị trí 1.

Câu 27. Viết function **int countLeave(TreeNode* root)** để đếm số lá có trong 1 cây nhị phân tổng quát.

Câu 28. Cho các khai báo sau:

```
struct StudentNode {
    int id;
    string name;
    StudentNode* next;
};

struct ClassNode {
    int id;
    string name;
    StudentNode* students;
    ClassNode* next;
};
```

Trong đó, cấu trúc StudentNode dùng để lưu trữ một node trong danh sách liên kết đơn chứa thông tin sinh viên của một lớp, ClassNode dùng để lưu trữ một node trong danh sách liên kết đơn chứa thông tin của các lớp đang quản lý trong một trường học. Thành phần students của ClassNode đại diện cho danh sách sinh viên của lớp đó.

Hãy định nghĩa hàm **bool addStudent(ClassNode* list, int classId, int studentId, string studentName)**, trong đó list là con trỏ trỏ đến phần tử đầu tiên trong danh sách lớp, classId, studentId, studentName lần lượt là mã số lớp, mã số sinh viên và tên của sinh viên. Hàm thực thực hiện chức năng sau:

- Thêm sinh viên vào đúng lớp có classId tương ứng và trả về true. Nếu không tồn tại classId đó, không thực hiện và trả về false.

- Nếu tồn tại studentId (trong toàn bộ các lớp) không thực hiện và trả về false.

Câu 29. Viết function **void printSmallerNearestNode(BSTNode* root, int key)** để tìm kiếm và in ra màn hình node duy nhất có giá trị nhỏ (hoặc bằng) hơn và gần nhất với giá trị **key** trong 1 BST (giá trị **key** vì thế có thể không tồn tại trong cây).

TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH

Họ và tên:.....
MSSV:.....

KIỂM TRA CUỐI KỲ (25/07/2020)

Môn thi: Cấu trúc dữ liệu và giải thuật Thời gian làm bài: 120 phút

☐ Không được phép dùng tài liệu ☒ Được phép dùng tài liệu giấy
Đề thi số: **3** **Đề thi gồm 6 trang.**

Chú ý: Sinh viên làm trực tiếp trên đề thi; mỗi câu hỏi trắc nghiệm chỉ chọn một đáp án đúng nhất. Các khai báo về thư viện, các hàm cơ bản và khai báo khác (nếu cần thiết) được xem như đầy đủ

I. TRẮC NGHIỆM (3 điểm)

Câu 1. 1 hash table có length 10 sử dụng hash function h(k) = k mod 10, và linear probling. Sau khi chèn 6 giá trị vào 1 empty hash table, thì table chứa giá trị như sau: [_, _, 42, 23, 34, 52, 46, 33, _, _] (index của table được đánh từ 0). Các key values được chèn theo thứ tự trước sau như thế nào?

(A) 42, 46, 33, 23, 34, 52 (B) 46, 42, 34, 52, 23, 33
(C) 34, 42, 23, 52, 33, 46 (D) 46, 34, 42, 23, 52, 33

Câu 2. Xét đoạn code sau, độ phức tạp Big-O:

```
int test = 0, i, j;
for (i = 0; i < n; i++):
    for (j = 0; j < i; j++):
        test = test + i * j;
```

(A) O(1) (B) O(n)
(C) O(n^2) (D) O(log n)

Câu 3. Con trỏ (pointers gồm front, rear) nào sẽ bị thay đổi khi insert 1 phần tử vào 1 queue đang có 1 phần tử:

(A) không có con trỏ nào thay đổi
(B) front (C) rear
(D) cả 2

Câu 4. Giả sử cần chuyển biểu thức trung tố sang hậu tố, cấu trúc dữ liệu nào phù hợp nhất được sử dụng để giải quyết bài toán này:

(A) Stack (B) Queue
(C) List (D) Tree

Câu 5. Thuật ngữ nào dùng để diễn tả giải thuật O(n):

(A) Cubic (B) Constant
(C) Linear (D) Logarithmic

Câu 6. Số node tối thiểu của 1 **full** binary tree có depth = 3 (cây chỉ có root xem như có depth = 0) là:

(A) 7 (B) 8
(C) 3 (D) 5
(E) 15

Câu 7. Dạng nào của List phù hợp để trả lời câu hỏi "Phần tử thứ i của list chứa gì?":

(A) Singly-linked lists hoặc Doubly-linked lists đều phù hợp
(B) Lists hiện thực bằng 1 array
(C) Singly-linked lists
(D) Doubly-linked lists (danh sách liên kết 2 chiều)

Câu 8. Xét giải thuật dùng stack để xác định sự cân bằng của 1 chuỗi các dấu đóng, mở ngoặc. Số phần tử trong stack tại một thời điểm (AT ANY ONE TIME) lớn nhất là bao nhiêu khi phân tích chuỗi sau: (()(())(()))

(A) 5 hoặc nhiều hơn (B) 4
(C) 1 (D) 2
(E) 3

Câu 9. Trong trường hợp tốt nhất, giải thuật sort nào hiệu quả nhất:

(A) Selection sort (B) Insertion sort
(C) Quick sort (D) Merge sort

Câu 10. Cho array A chứa n số nguyên có giá trị khác nhau từng đôi một. Giả sử muốn tìm giá trị nhỏ nhất trong A, trong trường hợp xấu nhất, giải pháp nào sau đây hiệu quả nhất:

(A) Tạo 1 MinHeap từ A, min được chứa trong root
(B) Sort A tăng dần, min là phần tử đầu tiên
(C) Lặp qua cả dãy tìm phần tử min
(D) Tạo 1 BST từ A, tìm phần tử min

Câu 11. 1 cây nhị phân khi duyệt NLR cho kết quả **14 2 1 3 11 10 7 30 40**, duyệt LRN cho kết quả **1 3 2 7 10 40 30 11 14**, duyệt LNR sẽ có kết quả là:

(A) 14 1 2 3 10 11 7 40 30 (B) 1 2 3 7 10 11 14 30 40
(C) 1 2 3 14 7 10 11 40 30 (D) 1 2 3 7 10 14 30 11 40

Câu 12. Chọn phát biểu sai

(A) Searching trên Heap có độ phức tạp O(log n)
(B) Linear Searching có độ phức tạp O(n)
(C) Binary Searching trên Ordered Array có độ phức tạp O(log n)
(D) Searching trên cây AVL có độ phức tạp O(ln n)

Câu 13. Trong trường hợp tốt nhất, giải thuật sort nào kém hiệu quả nhất:

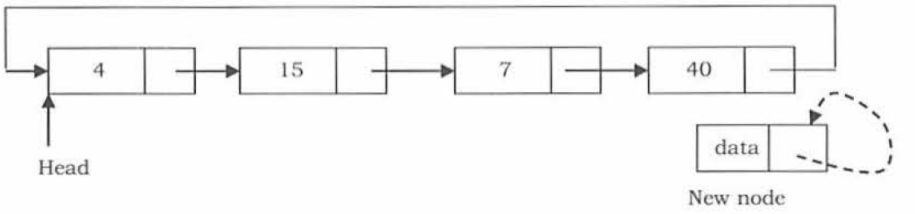
(A) Selection sort (B) Insertion sort
(C) Quick sort (D) Bubble sort

II. ĐIỀN KẾT QUẢ (4 điểm)

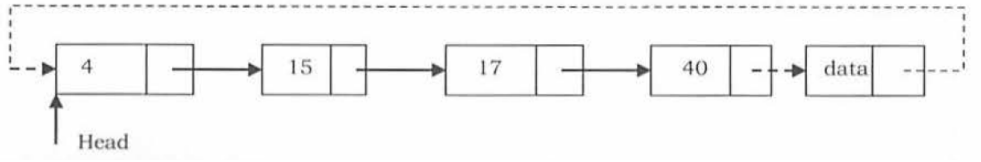
Câu 14. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], lần lượt insert các phần tử của A theo thứ tự index từ 0->7 vào 1 empty MaxHeap, hãy vẽ cây Heap cuối cùng được tạo ra.

Câu 15. Xóa lần lượt phần tử lớn nhất, lớn nhì, và lớn thứ 3 của MaxHeap ở câu trên. Hãy vẽ MaxHeap sau khi xóa.

Câu 16. Cho hình bên dưới:



Hãy viết các câu lệnh cần thiết để có thể chèn phần tử New node vào vị trí như hình sau, giả sử đã có 2 con trỏ Head và New node như hình:



Câu 17. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], sử dụng **insertion sort** sắp xếp A **giảm dần**, sau 4 bước sắp xếp (four complete passes) thì A = ?.

Câu 18. Cho array A = [21, 1, 26, 45, 29, 28, 2, 9, 16, 49, 39, 27, 43, 34, 46, 40], lần lượt insert các phần tử của A theo thứ tự index từ 0->15 vào 1 cây empty AVL, hãy vẽ cây AVL cuối cùng được tạo ra.

Câu 19. Xóa lần lượt 9, và 29 của cây AVL ở câu trên. Hãy vẽ cây AVL sau khi xóa.

Câu 20. Chèn các khóa 12, 18, 13, 2, 3, 23, 5 và 15 vào 1 empty hash table có size là 10, sử dụng h(k) = k mod 10 và linear probling. Hãy vẽ hash table kết quả (index từ 0).

Câu 21. Cho function sau:

```
int mystery_1(Node *tree = this.root):
    if(tree == nullptr):
        return 0;
    elif (tree.getLeft() == nullptr and tree.getRight() == nullptr):
        return 0;
    return 1 + mystery_1(tree.getLeft()) + mystery_1(tree.getRight());
```

Hãy cho biết công dụng của function này:

Câu 22. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], sử dụng **bubble sort** sắp xếp A **tăng dần**, sau 4 bước sắp xếp (four complete passes) thì A = ?.

Câu 23. Cho function sau:

```
int mystery_2(Node *tree = this.root):
    if(tree == nullptr)
        return 0;
    else if (tree.getLeft() == nullptr and tree.getRight() == nullptr):
        return 1;
    return 1 + mystery_2(tree.getLeft()) + mystery_2(tree.getRight());
```

Hãy cho biết công dụng của function này:

Câu 24. Cho array A = [21, 1, 26, 45, 29, 28, 2, 9, 16, 49, 39, 27, 43, 34, 46, 40], lần lượt insert các phần tử của A theo thứ tự index từ 0->15 vào 1 empty BST, hãy vẽ BST cuối cùng được tạo ra.

III. LẬP TRÌNH (5 điểm)

Câu 25. (1đ) (CODING STYLE)
Sinh viên sẽ có điểm câu này nếu làm ít nhất 1 câu trong các câu tiếp theo và có coding style tốt. Điểm được chấm theo 3 mức: 0, 0.5, hoặc 1.

Câu 26. Viết function **bool moveK2first(Node* head, int k)** để di chuyển node ở vị trí thứ k ra đầu danh sách. Giả sử phần tử đầu tiên nằm ở vị trí 1.

Câu 27. Viết function **int countLeave(TreeNode* root)** để đếm số lá có trong 1 cây nhị phân tổng quát.

Câu 28. Cho các khai báo sau:

```
struct StudentNode {
    int id;
    string name;
    StudentNode* next;
};

struct ClassNode {
    int id;
    string name;
    StudentNode* students;
    ClassNode* next;
};
```

Trong đó, cấu trúc StudentNode dùng để lưu trữ một node trong danh sách liên kết đơn chứa thông tin sinh viên của một lớp, ClassNode dùng để lưu trữ một node trong danh sách liên kết đơn chứa thông tin của các lớp đang quản lý trong một trường học. Thành phần students của ClassNode đại diện cho danh sách sinh viên của lớp đó.

Hãy định nghĩa hàm **bool addStudent(ClassNode* list, int classId, int studentId, string studentName)**, trong đó list là con trỏ trỏ đến phần tử đầu tiên trong danh sách lớp, classId, studentId, studentName lần lượt là mã số lớp, mã số sinh viên và tên của sinh viên. Hàm thực thực hiện chức năng sau:

- Thêm sinh viên vào đúng lớp có classId tương ứng và trả về true. Nếu không tồn tại classId đó, không thực hiện và trả về false.

- Nếu tồn tại studentId (trong toàn bộ các lớp) không thực hiện và trả về false.

Câu 29. Viết function **void printSmallerNearestNode(BSTNode* root, int key)** để tìm kiếm và in ra màn hình node duy nhất có giá trị nhỏ (hoặc bằng) hơn và gần nhất với giá trị **key** trong 1 BST (giá trị **key** vì thế có thể không tồn tại trong cây).

Chủ nhiệm Khoa/Bộ môn	Giảng viên ra đề
	ThS. Vương Bá Thịnh

TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH

KIỂM TRA CUỐI KỲ (25/07/2020)

Môn thi: Cấu trúc dữ liệu và giải thuật Thời gian làm bài: 120 phút

☐ Không được phép dùng tài liệu ☒ Được phép dùng tài liệu giấy
Đề thi số: 3

I. TRẮC NGHIỆM (3 điểm)

Câu 1. 1 hash table có length 10 sử dụng hash function h(k) = k mod 10, và linear probling. Sau khi chèn 6 giá trị vào 1 empty hash table, thì table chứa giá trị như sau: [_, _, 42, 23, 34, 52, 46, 33, _, _] (index của table được đánh từ 0). Các key values được chèn theo thứ tự trước sau như thế nào?

(A) 42, 46, 33, 23, 34, 52
(B) 46, 42, 34, 52, 23, 33
(C) 34, 42, 23, 52, 33, 46
[D] 46, 34, 42, 23, 52, 33

Câu 2. Xét đoạn code sau, độ phức tạp Big-O:

```
int test = 0, i, j;
for (i = 0; i < n; i++):
    for (j = 0; j < i; j++):
        test = test + i * j;
```

(A) O(1)
(B) O(n)
[C] O(n^2)
(D) O(log n)

Câu 3. Con trỏ (pointers gồm front, rear) nào sẽ bị thay đổi khi insert 1 phần tử vào 1 queue đang có 1 phần tử:

(A) không có con trỏ nào thay đổi
(B) front
[C] rear
(D) cả 2

Câu 4. Giả sử cần chuyển biểu thức trung tố sang hậu tố, cấu trúc dữ liệu nào phù hợp nhất được sử dụng để giải quyết bài toán này:

[A] Stack
(B) Queue
(C) List
(D) Tree

Câu 5. Thuật ngữ nào dùng để diễn tả giải thuật O(n):

(A) Cubic
(B) Constant
[C] Linear
(D) Logarithmic

Câu 6. Số node tối thiểu của 1 **full** binary tree có depth = 3 (cây chỉ có root xem như có depth = 0) là:

[A] 7
(B) 8
(C) 3
(D) 5
(E) 15

Câu 7. Dạng nào của List phù hợp để trả lời câu hỏi "Phần tử thứ i của list chứa gì?":

(A) Singly-linked lists hoặc Doubly-linked lists đều phù hợp
[B] Lists hiện thực bằng 1 array
(C) Singly-linked lists
(D) Doubly-linked lists (danh sách liên kết 2 chiều)

Câu 8. Xét giải thuật dùng stack để xác định sự cân bằng của 1 chuỗi các dấu đóng, mở ngoặc. Số phần tử trong stack tại một thời điểm (AT ANY ONE TIME) lớn nhất là bao nhiêu khi phân tích chuỗi sau: (()(())(()))

[A] 5 hoặc nhiều hơn
(B) 4
(C) 1
(D) 2
(E) 3

Câu 9. Trong trường hợp tốt nhất, giải thuật sort nào hiệu quả nhất:

(A) Selection sort
[B] Insertion sort
(C) Quick sort
(D) Merge sort

Câu 10. Cho array A chứa n số nguyên có giá trị khác nhau từng đôi một. Giả sử muốn tìm giá trị nhỏ nhất trong A, trong trường hợp xấu nhất, giải pháp nào sau đây hiệu quả nhất:

(A) Tạo 1 MinHeap từ A, min được chứa trong root
(B) Sort A tăng dần, min là phần tử đầu tiên
[C] Lặp qua cả dãy tìm phần tử min
(D) Tạo 1 BST từ A, tìm phần tử min

Câu 11. 1 cây nhị phân khi duyệt NLR cho kết quả **14 2 1 3 11 10 7 30 40**, duyệt LRN cho kết quả **1 3 2 7 10 40 30 11 14**, duyệt LNR sẽ có kết quả là:

(A) 14 1 2 3 10 11 7 40 30
(B) 1 2 3 7 10 11 14 30 40
[C] 1 2 3 14 7 10 11 40 30
(D) 1 2 3 7 10 14 30 11 40

Câu 12. Chọn phát biểu sai

[A] Searching trên Heap có độ phức tạp O(log n)
(B) Linear Searching có độ phức tạp O(n)
(C) Binary Searching trên Ordered Array có độ phức tạp O(log n)
(D) Searching trên cây AVL có độ phức tạp O(ln n)

Câu 13. Trong trường hợp tốt nhất, giải thuật sort nào kém hiệu quả nhất:

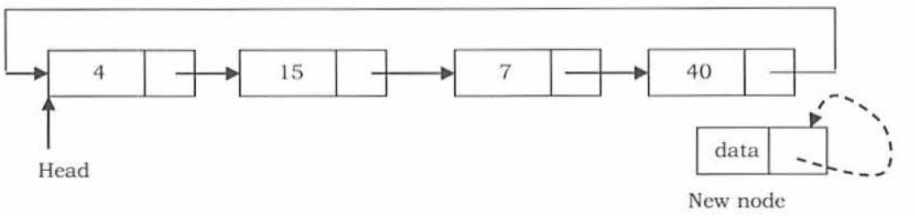
(A) Selection sort (B) Insertion sort
(C) Quick sort [D] Bubble sort

II. ĐIỀN KẾT QUẢ (4 điểm)

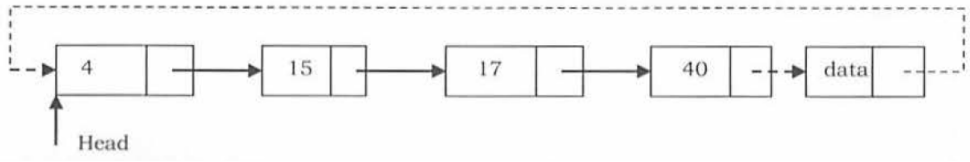
Câu 14. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], lần lượt insert các phần tử của A theo thứ tự index từ 0->7 vào 1 empty MaxHeap, hãy vẽ cây Heap cuối cùng được tạo ra.

Câu 15. Xóa lần lượt phần tử lớn nhất, lớn nhì, và lớn thứ 3 của MaxHeap ở câu trên. Hãy vẽ MaxHeap sau khi xóa.

Câu 16. Cho hình bên dưới:



Hãy viết các câu lệnh cần thiết để có thể chèn phần tử New node vào vị trí như hình sau, giả sử đã có 2 con trỏ Head và New node như hình:



Câu 17. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], sử dụng **insertion sort** sắp xếp A **giảm dần**, sau 4 bước sắp xếp (four complete passes) thì A = ?.

Câu 18. Cho array A = [21, 1, 26, 45, 29, 28, 2, 9, 16, 49, 39, 27, 43, 34, 46, 40], lần lượt insert các phần tử của A theo thứ tự index từ 0->15 vào 1 cây empty AVL, hãy vẽ cây AVL cuối cùng được tạo ra.

Câu 19. Xóa lần lượt 9, và 29 của cây AVL ở câu trên. Hãy vẽ cây AVL sau khi xóa.

Câu 20. Chèn các khóa 12, 18, 13, 2, 3, 23, 5 và 15 vào 1 empty hash table có size là 10, sử dụng h(k) = k mod 10 và linear probling. Hãy vẽ hash table kết quả (index từ 0).

Câu 21. Cho function sau:

```
int mystery_1(Node *tree = this.root):
    if(tree == nullptr):
        return 0;
    elif (tree.getLeft() == nullptr and tree.getRight() == nullptr):
        return 0;
    return 1 + mystery_1(tree.getLeft()) + mystery_1(tree.getRight());
```

Hãy cho biết công dụng của function này:

Câu 22. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], sử dụng **bubble sort** sắp xếp A **tăng dần**, sau 4 bước sắp xếp (four complete passes) thì A = ?.

Câu 23. Cho function sau:

```
int mystery_2(Node *tree = this.root):
    if(tree == nullptr)
        return 0;
    else if (tree.getLeft() == nullptr and tree.getRight() == nullptr):
        return 1;
    return 1 + mystery_2(tree.getLeft()) + mystery_2(tree.getRight());
```

Hãy cho biết công dụng của function này:

Câu 24. Cho array A = [21, 1, 26, 45, 29, 28, 2, 9, 16, 49, 39, 27, 43, 34, 46, 40], lần lượt insert các phần tử của A theo thứ tự index từ 0->15 vào 1 empty BST, hãy vẽ BST cuối cùng được tạo ra.

III. LẬP TRÌNH (5 điểm)

Câu 25. (1đ) (CODING STYLE)
Sinh viên sẽ có điểm câu này nếu làm ít nhất 1 câu trong các câu tiếp theo và có coding style tốt. Điểm được chấm theo 3 mức: 0, 0.5, hoặc 1.

Câu 26. Viết function **bool moveK2first(Node* head, int k)** để di chuyển node ở vị trí thứ k ra đầu danh sách. Giả sử phần tử đầu tiên nằm ở vị trí 1.

Câu 27. Viết function **int countLeave(TreeNode* root)** để đếm số lá có trong 1 cây nhị phân tổng quát.

Câu 28. Cho các khai báo sau:

```
struct StudentNode {
    int id;
    string name;
    StudentNode* next;
};

struct ClassNode {
    int id;
    string name;
    StudentNode* students;
    ClassNode* next;
};
```

Trong đó, cấu trúc StudentNode dùng để lưu trữ một node trong danh sách liên kết đơn chứa thông tin sinh viên của một lớp, ClassNode dùng để lưu trữ một node trong danh sách liên kết đơn chứa thông tin của các lớp đang quản lý trong một trường học. Thành phần students của ClassNode đại diện cho danh sách sinh viên của lớp đó.

Hãy định nghĩa hàm **bool addStudent(ClassNode* list, int classId, int studentId, string studentName)**, trong đó list là con trỏ trỏ đến phần tử đầu tiên trong danh sách lớp, classId, studentId, studentName lần lượt là mã số lớp, mã số sinh viên và tên của sinh viên. Hàm thực thực hiện chức năng sau:

- Thêm sinh viên vào đúng lớp có classId tương ứng và trả về true. Nếu không tồn tại classId đó, không thực hiện và trả về false.

- Nếu tồn tại studentId (trong toàn bộ các lớp) không thực hiện và trả về false.

Câu 29. Viết function **void printSmallerNearestNode(BSTNode* root, int key)** để tìm kiếm và in ra màn hình node duy nhất có giá trị nhỏ (hoặc bằng) hơn và gần nhất với giá trị **key** trong 1 BST (giá trị **key** vì thế có thể không tồn tại trong cây).

TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH

Họ và tên:.....
MSSV:.....

KIỂM TRA CUỐI KỲ (25/07/2020)

Môn thi: Cấu trúc dữ liệu và giải thuật Thời gian làm bài: 120 phút

☐ Không được phép dùng tài liệu ☒ Được phép dùng tài liệu giấy
Đề thi số: **5** **Đề thi gồm 6 trang.**

Chú ý: Sinh viên làm trực tiếp trên đề thi; mỗi câu hỏi trắc nghiệm chỉ chọn một đáp án đúng nhất. Các khai báo về thư viện, các hàm cơ bản và khai báo khác (nếu cần thiết) được xem như đầy đủ

I. TRẮC NGHIỆM (3 điểm)

Câu 1. Giả sử cần chuyển biểu thức trung tố sang hậu tố, cấu trúc dữ liệu nào phù hợp nhất được sử dụng để giải quyết bài toán này:

(A) Queue (B) Stack
(C) List (D) Tree

Câu 2. Xét đoạn code sau, độ phức tạp Big-O:

```
int test = 0, i, j;
for (i = 0; i < n; i++):
    for (j = 0; j < i; j++):
        test = test + i * j;
```

(A) O(n) (B) O(1)
(C) O(n^2) (D) O(log n)

Câu 3. Trong trường hợp tốt nhất, giải thuật sort nào hiệu quả nhất:

(A) Insertion sort (B) Selection sort
(C) Quick sort (D) Merge sort

Câu 4. Dạng nào của List phù hợp để trả lời câu hỏi "Phần tử thứ i của list chứa gì?":

(A) Lists hiện thực bằng 1 array
(B) Singly-linked lists hoặc Doubly-linked lists đều phù hợp
(C) Singly-linked lists
(D) Doubly-linked lists (danh sách liên kết 2 chiều)

Câu 5. Con trỏ (pointers gồm front, rear) nào sẽ bị thay đổi khi insert 1 phần tử vào 1 queue đang có 1 phần tử:

(A) front
(B) không có con trỏ nào thay đổi
(C) rear (D) cả 2

Câu 6. Cho array A chứa n số nguyên có giá trị khác nhau từng đôi một. Giả sử muốn tìm giá trị nhỏ nhất trong A, trong trường hợp xấu nhất, giải pháp nào sau đây hiệu quả nhất:

(A) Sort A tăng dần, min là phần tử đầu tiên
(B) Tạo 1 MinHeap từ A, min được chứa trong root
(C) Lặp qua cả dãy tìm phần tử min
(D) Tạo 1 BST từ A, tìm phần tử min

Câu 7. 1 hash table có length 10 sử dụng hash function h(k) = k mod 10, và linear probling. Sau khi chèn 6 giá trị vào 1 empty hash table, thì table chứa giá trị như sau: [_, _, 42, 23, 34, 52, 46, 33, _, _] (index của table được đánh từ 0). Các key values được chèn theo thứ tự trước sau như thế nào?

(A) 46, 42, 34, 52, 23, 33 (B) 42, 46, 33, 23, 34, 52
(C) 34, 42, 23, 52, 33, 46 (D) 46, 34, 42, 23, 52, 33

Câu 8. Xét giải thuật dùng stack để xác định sự cân bằng của 1 chuỗi các dấu đóng, mở ngoặc. Số phần tử trong stack tại một thời điểm (AT ANY ONE TIME) lớn nhất là bao nhiêu khi phân tích chuỗi sau: (()(())(()))

(A) 5 hoặc nhiều hơn (B) 1
(C) 4 (D) 2
(E) 3

Câu 9. 1 cây nhị phân khi duyệt NLR cho kết quả **14 2 1 3 11 10 7 30 40**, duyệt LRN cho kết quả **1 3 2 7 10 40 30 11 14**, duyệt LNR sẽ có kết quả là:

(A) 1 2 3 7 10 11 14 30 40 (B) 14 1 2 3 10 11 7 40 30
(C) 1 2 3 14 7 10 11 40 30 (D) 1 2 3 7 10 14 30 11 40

Câu 10. Số node tối thiểu của 1 **full** binary tree có depth = 3 (cây chỉ có root xem như có depth = 0) là:

(A) 7 (B) 3
(C) 8 (D) 5
(E) 15

Câu 11. Chọn phát biểu sai

(A) Linear Searching có độ phức tạp O(n)
(B) Searching trên Heap có độ phức tạp O(log n)
(C) Binary Searching trên Ordered Array có độ phức tạp O(log n)
(D) Searching trên cây AVL có độ phức tạp O(ln n)

Câu 12. Trong trường hợp tốt nhất, giải thuật sort nào kém hiệu quả nhất:

(A) Insertion sort (B) Selection sort
(C) Quick sort (D) Bubble sort

Câu 13. Thuật ngữ nào dùng để diễn tả giải thuật O(n):

(A) Constant (B) Cubic
(C) Linear (D) Logarithmic

II. ĐIỀN KẾT QUẢ (4 điểm)

Câu 14. Cho function sau:

```
int mystery_1(Node *tree = this.root):
    if(tree == nullptr):
        return 0;
    elif (tree.getLeft() == nullptr and tree.getRight() == nullptr):
        return 0;
    return 1 + mystery_1(tree.getLeft()) + mystery_1(tree.getRight());
```

Hãy cho biết công dụng của function này:

Câu 15. Chèn các khóa 12, 18, 13, 2, 3, 23, 5 và 15 vào 1 empty hash table có size là 10, sử dụng h(k) = k mod 10 và linear probling. Hãy vẽ hash table kết quả (index từ 0).

Câu 16. Cho function sau:

```
int mystery_2(Node *tree = this.root):
    if(tree == nullptr)
        return 0;
    else if (tree.getLeft() == nullptr and tree.getRight() == nullptr):
        return 1;
    return 1 + mystery_2(tree.getLeft()) + mystery_2(tree.getRight());
```

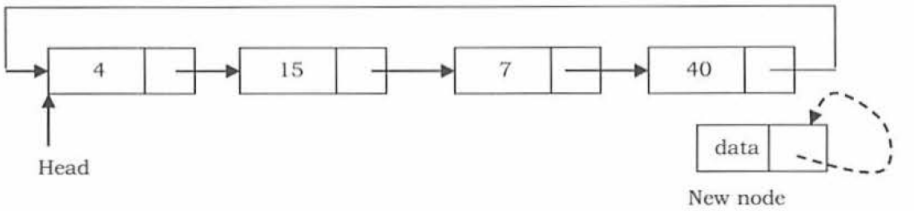
Hãy cho biết công dụng của function này:

Câu 17. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], sử dụng **bubble sort** sắp xếp A **tăng dần**, sau 4 bước sắp xếp (four complete passes) thì A = ?.

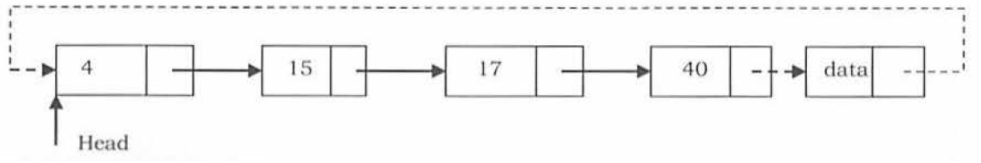
Câu 18. Cho array A = [21, 1, 26, 45, 29, 28, 2, 9, 16, 49, 39, 27, 43, 34, 46, 40], lần lượt insert các phần tử của A theo thứ tự index từ 0->15 vào 1 cây empty AVL, hãy vẽ cây AVL cuối cùng được tạo ra.

Câu 19. Xóa lần lượt 9, và 29 của cây AVL ở câu trên. Hãy vẽ cây AVL sau khi xóa.

Câu 20. Cho hình bên dưới:



Hãy viết các câu lệnh cần thiết để có thể chèn phần tử New node vào vị trí như hình sau, giả sử đã có 2 con trỏ Head và New node như hình:



Câu 21. Cho array A = [21, 1, 26, 45, 29, 28, 2, 9, 16, 49, 39, 27, 43, 34, 46, 40], lần lượt insert các phần tử của A theo thứ tự index từ 0->15 vào 1 empty BST, hãy vẽ BST cuối cùng được tạo ra.

Câu 22. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], lần lượt insert các phần tử của A theo thứ tự index từ 0->7 vào 1 empty MaxHeap, hãy vẽ cây Heap cuối cùng được tạo ra.

Câu 23. Xóa lần lượt phần tử lớn nhất, lớn nhì, và lớn thứ 3 của MaxHeap ở câu trên. Hãy vẽ MaxHeap sau khi xóa.

Câu 24. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], sử dụng **insertion sort** sắp xếp A **giảm dần**, sau 4 bước sắp xếp (four complete passes) thì A = ?.

III. LẬP TRÌNH (5 điểm)

Câu 25. (1đ) (CODING STYLE)
Sinh viên sẽ có điểm câu này nếu làm ít nhất 1 câu trong các câu tiếp theo và có coding style tốt. Điểm được chấm theo 3 mức: 0, 0.5, hoặc 1.

Câu 26. Viết function **bool moveK2first(Node* head, int k)** để di chuyển node ở vị trí thứ k ra đầu danh sách. Giả sử phần tử đầu tiên nằm ở vị trí 1.

Câu 27. Viết function **int countLeave(TreeNode* root)** để đếm số lá có trong 1 cây nhị phân tổng quát.

Câu 28. Cho các khai báo sau:

```
struct StudentNode {
    int id;
    string name;
    StudentNode* next;
};

struct ClassNode {
    int id;
    string name;
    StudentNode* students;
    ClassNode* next;
};
```

Trong đó, cấu trúc StudentNode dùng để lưu trữ một node trong danh sách liên kết đơn chứa thông tin sinh viên của một lớp, ClassNode dùng để lưu trữ một node trong danh sách liên kết đơn chứa thông tin của các lớp đang quản lý trong một trường học. Thành phần students của ClassNode đại diện cho danh sách sinh viên của lớp đó.

Hãy định nghĩa hàm **bool addStudent(ClassNode* list, int classId, int studentId, string studentName)**, trong đó list là con trỏ trỏ đến phần tử đầu tiên trong danh sách lớp, classId, studentId, studentName lần lượt là mã số lớp, mã số sinh viên và tên của sinh viên. Hàm thực thực hiện chức năng sau:

- Thêm sinh viên vào đúng lớp có classId tương ứng và trả về true. Nếu không tồn tại classId đó, không thực hiện và trả về false.

- Nếu tồn tại studentId (trong toàn bộ các lớp) không thực hiện và trả về false.

Câu 29. Viết function **void printSmallerNearestNode(BSTNode* root, int key)** để tìm kiếm và in ra màn hình node duy nhất có giá trị nhỏ (hoặc bằng) hơn và gần nhất với giá trị **key** trong 1 BST (giá trị **key** vì thế có thể không tồn tại trong cây).

Chủ nhiệm Khoa/Bộ môn	Giảng viên ra đề
	ThS. Vương Bá Thịnh

TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH

KIỂM TRA CUỐI KỲ (25/07/2020)

Môn thi: Cấu trúc dữ liệu và giải thuật Thời gian làm bài: 120 phút

☐ Không được phép dùng tài liệu ☒ Được phép dùng tài liệu giấy
Đề thi số: 5

I. TRẮC NGHIỆM (3 điểm)

Câu 1. Giả sử cần chuyển biểu thức trung tố sang hậu tố, cấu trúc dữ liệu nào phù hợp nhất được sử dụng để giải quyết bài toán này:

(A) Queue
[B] Stack
(C) List
(D) Tree

Câu 2. Xét đoạn code sau, độ phức tạp Big-O:

```
int test = 0, i, j;
for (i = 0; i < n; i++):
    for (j = 0; j < i; j++):
        test = test + i * j;
```

(A) O(n)
(B) O(1)
[C] O(n^2)
(D) O(log n)

Câu 3. Trong trường hợp tốt nhất, giải thuật sort nào hiệu quả nhất:

[A] Insertion sort
(B) Selection sort
(C) Quick sort
(D) Merge sort

Câu 4. Dạng nào của List phù hợp để trả lời câu hỏi "Phần tử thứ i của list chứa gì?":

[A] Lists hiện thực bằng 1 array
(B) Singly-linked lists hoặc Doubly-linked lists đều phù hợp
(C) Singly-linked lists
(D) Doubly-linked lists (danh sách liên kết 2 chiều)

Câu 5. Con trỏ (pointers gồm front, rear) nào sẽ bị thay đổi khi insert 1 phần tử vào 1 queue đang có 1 phần tử:

(A) front
(B) không có con trỏ nào thay đổi
[C] rear
(D) cả 2

Câu 6. Cho array A chứa n số nguyên có giá trị khác nhau từng đôi một. Giả sử muốn tìm giá trị nhỏ nhất trong A, trong trường hợp xấu nhất, giải pháp nào sau đây hiệu quả nhất:

(A) Sort A tăng dần, min là phần tử đầu tiên
(B) Tạo 1 MinHeap từ A, min được chứa trong root
[C] Lặp qua cả dãy tìm phần tử min
(D) Tạo 1 BST từ A, tìm phần tử min

Câu 7. 1 hash table có length 10 sử dụng hash function h(k) = k mod 10, và linear probing. Sau khi chèn 6 giá trị vào 1 empty hash table, thì table chứa giá trị như sau: [_, _, 42, 23, 34, 52, 46, 33, _, _] (index của table được đánh từ 0). Các key values được chèn theo thứ tự trước sau như thế nào?

(A) 46, 42, 34, 52, 23, 33
(B) 42, 46, 33, 23, 34, 52
(C) 34, 42, 23, 52, 33, 46
[D] 46, 34, 42, 23, 52, 33

Câu 8. Xét giải thuật dùng stack để xác định sự cân bằng của 1 chuỗi các dấu đóng, mở ngoặc. Số phần tử trong stack tại một thời điểm (AT ANY ONE TIME) lớn nhất là bao nhiêu khi phân tích chuỗi sau: (()(())(()))

[A] 5 hoặc nhiều hơn
(B) 1
(C) 4
(D) 2
(E) 3

Câu 9. 1 cây nhị phân khi duyệt NLR cho kết quả **14 2 1 3 11 10 7 30 40**, duyệt LRN cho kết quả **1 3 2 7 10 40 30 11 14**, duyệt LNR sẽ có kết quả là:

(A) 1 2 3 7 10 11 14 30 40
(B) 14 1 2 3 10 11 7 40 30
[C] 1 2 3 14 7 10 11 40 30
(D) 1 2 3 7 10 14 30 11 40

Câu 10. Số node tối thiểu của 1 **full** binary tree có depth = 3 (cây chỉ có root xem như có depth = 0) là:

[A] 7
(B) 3
(C) 8
(D) 5
(E) 15

Câu 11. Chọn phát biểu sai

(A) Linear Searching có độ phức tạp O(n)
[B] Searching trên Heap có độ phức tạp O(log n)
(C) Binary Searching trên Ordered Array có độ phức tạp O(log n)
(D) Searching trên cây AVL có độ phức tạp O(ln n)

Câu 12. Trong trường hợp tốt nhất, giải thuật sort nào kém hiệu quả nhất:

(A) Insertion sort
(B) Selection sort
(C) Quick sort
[D] Bubble sort

Câu 13. Thuật ngữ nào dùng để diễn tả giải thuật O(n):

(A) Constant (B) Cubic
[C] Linear (D) Logarithmic

II. ĐIỀN KẾT QUẢ (4 điểm)

Câu 14. Cho function sau:

```
int mystery_1(Node *tree = this.root):
    if(tree == nullptr):
        return 0;
    elif (tree.getLeft() == nullptr and tree.getRight() == nullptr):
        return 0;
    return 1 + mystery_1(tree.getLeft()) + mystery_1(tree.getRight());
```

Hãy cho biết công dụng của function này:

Câu 15. Chèn các khóa 12, 18, 13, 2, 3, 23, 5 và 15 vào 1 empty hash table có size là 10, sử dụng h(k) = k mod 10 và linear probling. Hãy vẽ hash table kết quả (index từ 0).

Câu 16. Cho function sau:

```
int mystery_2(Node *tree = this.root):
    if(tree == nullptr)
        return 0;
    else if (tree.getLeft() == nullptr and tree.getRight() == nullptr):
        return 1;
    return 1 + mystery_2(tree.getLeft()) + mystery_2(tree.getRight());
```

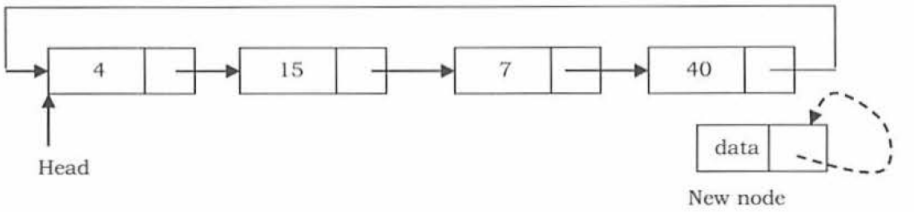
Hãy cho biết công dụng của function này:

Câu 17. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], sử dụng **bubble sort** sắp xếp A **tăng dần**, sau 4 bước sắp xếp (four complete passes) thì A = ?.

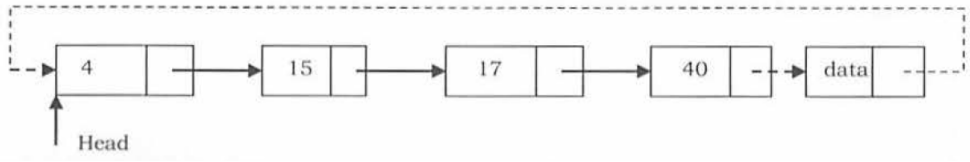
Câu 18. Cho array A = [21, 1, 26, 45, 29, 28, 2, 9, 16, 49, 39, 27, 43, 34, 46, 40], lần lượt insert các phần tử của A theo thứ tự index từ 0->15 vào 1 cây empty AVL, hãy vẽ cây AVL cuối cùng được tạo ra.

Câu 19. Xóa lần lượt 9, và 29 của cây AVL ở câu trên. Hãy vẽ cây AVL sau khi xóa.

Câu 20. Cho hình bên dưới:



Hãy viết các câu lệnh cần thiết để có thể chèn phần tử New node vào vị trí như hình sau, giả sử đã có 2 con trỏ Head và New node như hình:



Câu 21. Cho array A = [21, 1, 26, 45, 29, 28, 2, 9, 16, 49, 39, 27, 43, 34, 46, 40], lần lượt insert các phần tử của A theo thứ tự index từ 0->15 vào 1 empty BST, hãy vẽ BST cuối cùng được tạo ra.

Câu 22. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], lần lượt insert các phần tử của A theo thứ tự index từ 0->7 vào 1 empty MaxHeap, hãy vẽ cây Heap cuối cùng được tạo ra.

Câu 23. Xóa lần lượt phần tử lớn nhất, lớn nhì, và lớn thứ 3 của MaxHeap ở câu trên. Hãy vẽ MaxHeap sau khi xóa.

Câu 24. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], sử dụng **insertion sort** sắp xếp A **giảm dần**, sau 4 bước sắp xếp (four complete passes) thì A = ?.

III. LẬP TRÌNH (5 điểm)

Câu 25. (1đ) (CODING STYLE)
Sinh viên sẽ có điểm câu này nếu làm ít nhất 1 câu trong các câu tiếp theo và có coding style tốt. Điểm được chấm theo 3 mức: 0, 0.5, hoặc 1.

Câu 26. Viết function **bool moveK2first(Node* head, int k)** để di chuyển node ở vị trí thứ k ra đầu danh sách. Giả sử phần tử đầu tiên nằm ở vị trí 1.

Câu 27. Viết function **int countLeave(TreeNode* root)** để đếm số lá có trong 1 cây nhị phân tổng quát.

Câu 28. Cho các khai báo sau:

```
struct StudentNode {
    int id;
    string name;
    StudentNode* next;
};

struct ClassNode {
    int id;
    string name;
    StudentNode* students;
    ClassNode* next;
};
```

Trong đó, cấu trúc StudentNode dùng để lưu trữ một node trong danh sách liên kết đơn chứa thông tin sinh viên của một lớp, ClassNode dùng để lưu trữ một node trong danh sách liên kết đơn chứa thông tin của các lớp đang quản lý trong một trường học. Thành phần students của ClassNode đại diện cho danh sách sinh viên của lớp đó.

Hãy định nghĩa hàm **bool addStudent(ClassNode* list, int classId, int studentId, string studentName)**, trong đó list là con trỏ trỏ đến phần tử đầu tiên trong danh sách lớp, classId, studentId, studentName lần lượt là mã số lớp, mã số sinh viên và tên của sinh viên. Hàm thực thực hiện chức năng sau:

- Thêm sinh viên vào đúng lớp có classId tương ứng và trả về true. Nếu không tồn tại classId đó, không thực hiện và trả về false.

- Nếu tồn tại studentId (trong toàn bộ các lớp) không thực hiện và trả về false.

Câu 29. Viết function **void printSmallerNearestNode(BSTNode* root, int key)** để tìm kiếm và in ra màn hình node duy nhất có giá trị nhỏ (hoặc bằng) hơn và gần nhất với giá trị **key** trong 1 BST (giá trị **key** vì thế có thể không tồn tại trong cây).

TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH

Họ và tên:.....
MSSV:.....

KIỂM TRA CUỐI KỲ (25/07/2020)

Môn thi: Cấu trúc dữ liệu và giải thuật Thời gian làm bài: 120 phút

☐ Không được phép dùng tài liệu ☒ Được phép dùng tài liệu giấy
Đề thi số: **7** **Đề thi gồm 6 trang.**

Chú ý: Sinh viên làm trực tiếp trên đề thi; mỗi câu hỏi trắc nghiệm chỉ chọn một đáp án đúng nhất. Các khai báo về thư viện, các hàm cơ bản và khai báo khác (nếu cần thiết) được xem như đầy đủ

I. TRẮC NGHIỆM (3 điểm)

Câu 1. Xét đoạn code sau, độ phức tạp Big-O:

```
int test = 0, i, j;
for (i = 0; i < n; i++):
    for (j = 0; j < i; j++):
        test = test + i * j;
```

(A) O(n) (B) O(log n)
(C) O(n^2) (D) O(1)

Câu 2. Xét giải thuật dùng stack để xác định sự cân bằng của 1 chuỗi các dấu đóng, mở ngoặc. Số phần tử trong stack tại một thời điểm (AT ANY ONE TIME) lớn nhất là bao nhiêu khi phân tích chuỗi sau: (()(())(()))

(A) 5 hoặc nhiều hơn (B) 1
(C) 2 (D) 4
(E) 3

Câu 3. Chọn phát biểu sai

(A) Linear Searching có độ phức tạp O(n)
(B) Searching trên cây AVL có độ phức tạp O(ln n)
(C) Binary Searching trên Ordered Array có độ phức tạp O(log n)
(D) Searching trên Heap có độ phức tạp O(log n)

Câu 4. Trong trường hợp tốt nhất, giải thuật sort nào hiệu quả nhất:

(A) Insertion sort (B) Merge sort
(C) Quick sort (D) Selection sort

Câu 5. Số node tối thiểu của 1 **full** binary tree có depth = 3 (cây chỉ có root xem như có depth = 0) là:

(A) 7 (B) 3
(C) 5 (D) 8
(E) 15

Câu 6. Thuật ngữ nào dùng để diễn tả giải thuật O(n):

(A) Constant (B) Logarithmic
(C) Linear (D) Cubic

Câu 7. Trong trường hợp tốt nhất, giải thuật sort nào kém hiệu quả nhất:

(A) Insertion sort (B) Bubble sort
(C) Quick sort (D) Selection sort

Câu 8. Giả sử cần chuyển biểu thức trung tố sang hậu tố, cấu trúc dữ liệu nào phù hợp nhất được sử dụng để giải quyết bài toán này:

(A) Queue (B) Tree
(C) List (D) Stack

Câu 9. 1 hash table có length 10 sử dụng hash function h(k) = k mod 10, và linear probling. Sau khi chèn 6 giá trị vào 1 empty hash table, thì table chứa giá trị như sau: [_, _, 42, 23, 34, 52, 46, 33, _, _] (index của table được đánh từ 0). Các key values được chèn theo thứ tự trước sau như thế nào?

(A) 46, 42, 34, 52, 23, 33 (B) 46, 34, 42, 23, 52, 33
(C) 34, 42, 23, 52, 33, 46 (D) 42, 46, 33, 23, 34, 52

Câu 10. Con trỏ (pointers gồm front, rear) nào sẽ bị thay đổi khi insert 1 phần tử vào 1 queue đang có 1 phần tử:

(A) front (B) cả 2
(C) rear
(D) không có con trỏ nào thay đổi

Câu 11. Cho array A chứa n số nguyên có giá trị khác nhau từng đôi một. Giả sử muốn tìm giá trị nhỏ nhất trong A, trong trường hợp xấu nhất, giải pháp nào sau đây hiệu quả nhất:

(A) Sort A tăng dần, min là phần tử đầu tiên
(B) Tạo 1 BST từ A, tìm phần tử min
(C) Lặp qua cả dãy tìm phần tử min
(D) Tạo 1 MinHeap từ A, min được chứa trong root

Câu 12. Dạng nào của List phù hợp để trả lời câu hỏi "Phần tử thứ i của list chứa gì?":

(A) Lists hiện thực bằng 1 array
(B) Doubly-linked lists (danh sách liên kết 2 chiều)
(C) Singly-linked lists
(D) Singly-linked lists hoặc Doubly-linked lists đều phù hợp

Câu 13. 1 cây nhị phân khi duyệt NLR cho kết quả **14 2 1 3 11 10 7 30 40**, duyệt LRN cho kết quả **1 3 2 7 10 40 30 11 14**, duyệt LNR sẽ có kết quả là:

(A) 1 2 3 7 10 11 14 30 40 (B) 1 2 3 7 10 14 30 11 40
(C) 1 2 3 14 7 10 11 40 30 (D) 14 1 2 3 10 11 7 40 30

II. ĐIỀN KẾT QUẢ (4 điểm)

Câu 14. Cho function sau:

```
int mystery_2(Node *tree = this.root):
    if(tree == nullptr)
        return 0;
    else if (tree.getLeft() == nullptr and tree.getRight() == nullptr):
        return 1;
    return 1 + mystery_2(tree.getLeft()) + mystery_2(tree.getRight());
```

Hãy cho biết công dụng của function này:

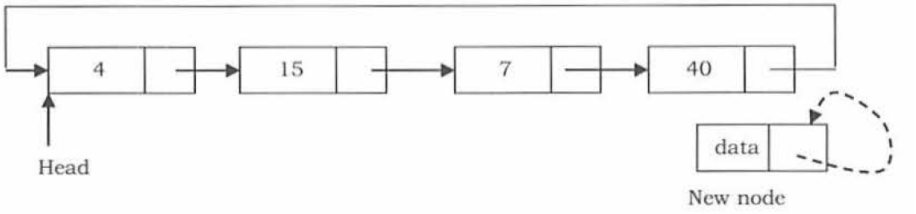
Câu 15. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], sử dụng **bubble sort** sắp xếp A **tăng dần**, sau 4 bước sắp xếp (four complete passes) thì A = ?.

Câu 16. Cho function sau:

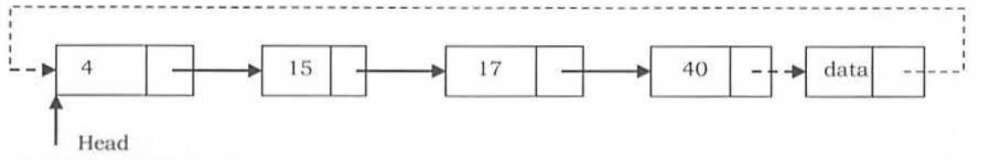
```
int mystery_1(Node *tree = this.root):
    if(tree == nullptr):
        return 0;
    elif (tree.getLeft() == nullptr and tree.getRight() == nullptr):
        return 0;
    return 1 + mystery_1(tree.getLeft()) + mystery_1(tree.getRight());
```

Hãy cho biết công dụng của function này:

Câu 17. Cho hình bên dưới:



Hãy viết các câu lệnh cần thiết để có thể chèn phần tử New node vào vị trí như hình sau, giả sử đã có 2 con trỏ Head và New node như hình:



Câu 18. Chèn các khóa 12, 18, 13, 2, 3, 23, 5 và 15 vào 1 empty hash table có size là 10, sử dụng h(k) = k mod 10 và linear probling. Hãy vẽ hash table kết quả (index từ 0).

Câu 19. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], sử dụng **insertion sort** sắp xếp A **giảm dần**, sau 4 bước sắp xếp (four complete passes) thì A = ?.

Câu 20. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], lần lượt insert các phần tử của A theo thứ tự index từ 0->7 vào 1 empty MaxHeap, hãy vẽ cây Heap cuối cùng được tạo ra.

Câu 21. Xóa lần lượt phần tử lớn nhất, lớn nhì, và lớn thứ 3 của MaxHeap ở câu trên. Hãy vẽ MaxHeap sau khi xóa.

Câu 22. Cho array A = [21, 1, 26, 45, 29, 28, 2, 9, 16, 49, 39, 27, 43, 34, 46, 40], lần lượt insert các phần tử của A theo thứ tự index từ 0->15 vào 1 cây empty AVL, hãy vẽ cây AVL cuối cùng được tạo ra.

Câu 23. Xóa lần lượt 9, và 29 của cây AVL ở câu trên. Hãy vẽ cây AVL sau khi xóa.

Câu 24. Cho array A = [21, 1, 26, 45, 29, 28, 2, 9, 16, 49, 39, 27, 43, 34, 46, 40], lần lượt insert các phần tử của A theo thứ tự index từ 0->15 vào 1 empty BST, hãy vẽ BST cuối cùng được tạo ra.

III. LẬP TRÌNH (5 điểm)

Câu 25. (1đ) (CODING STYLE)
Sinh viên sẽ có điểm câu này nếu làm ít nhất 1 câu trong các câu tiếp theo và có coding style tốt. Điểm được chấm theo 3 mức: 0, 0.5, hoặc 1.

Câu 26. Viết function **bool moveK2first(Node* head, int k)** để di chuyển node ở vị trí thứ k ra đầu danh sách. Giả sử phần tử đầu tiên nằm ở vị trí 1.

Câu 27. Viết function **int countLeave(TreeNode* root)** để đếm số lá có trong 1 cây nhị phân tổng quát.

Câu 28. Cho các khai báo sau:

```
struct StudentNode {
    int id;
    string name;
    StudentNode* next;
};

struct ClassNode {
    int id;
    string name;
    StudentNode* students;
    ClassNode* next;
};
```

Trong đó, cấu trúc StudentNode dùng để lưu trữ một node trong danh sách liên kết đơn chứa thông tin sinh viên của một lớp, ClassNode dùng để lưu trữ một node trong danh sách liên kết đơn chứa thông tin của các lớp đang quản lý trong một trường học. Thành phần students của ClassNode đại diện cho danh sách sinh viên của lớp đó.

Hãy định nghĩa hàm **bool addStudent(ClassNode* list, int classId, int studentId, string studentName)**, trong đó list là con trỏ trỏ đến phần tử đầu tiên trong danh sách lớp, classId, studentId, studentName lần lượt là mã số lớp, mã số sinh viên và tên của sinh viên. Hàm thực thực hiện chức năng sau:

- Thêm sinh viên vào đúng lớp có classId tương ứng và trả về true. Nếu không tồn tại classId đó, không thực hiện và trả về false.

- Nếu tồn tại studentId (trong toàn bộ các lớp) không thực hiện và trả về false.

Câu 29. Viết function **void printSmallerNearestNode(BSTNode* root, int key)** để tìm kiếm và in ra màn hình node duy nhất có giá trị nhỏ (hoặc bằng) hơn và gần nhất với giá trị **key** trong 1 BST (giá trị **key** vì thế có thể không tồn tại trong cây).

Chủ nhiệm Khoa/Bộ môn	Giảng viên ra đề
	ThS. Vương Bá Thịnh

TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH

KIỂM TRA CUỐI KỲ (25/07/2020)

Môn thi: Cấu trúc dữ liệu và giải thuật Thời gian làm bài: 120 phút

☐ Không được phép dùng tài liệu ☒ Được phép dùng tài liệu giấy
Đề thi số: 7

I. TRẮC NGHIỆM (3 điểm)

Câu 1. Xét đoạn code sau, độ phức tạp Big-O:

```
int test = 0, i, j;
for (i = 0; i < n; i++):
    for (j = 0; j < i; j++):
        test = test + i * j;
```

(A) O(n) (B) O(log n)
[C] O(n^2) (D) O(1)

Câu 2. Xét giải thuật dùng stack để xác định sự cân bằng của 1 chuỗi các dấu đóng, mở ngoặc. Số phần tử trong stack tại một thời điểm (AT ANY ONE TIME) lớn nhất là bao nhiêu khi phân tích chuỗi sau: ((()(()(())))

[A] 5 hoặc nhiều hơn (B) 1
(C) 2 (D) 4
(E) 3

Câu 3. Chọn phát biểu sai

(A) Linear Searching có độ phức tạp O(n)
(B) Searching trên cây AVL có độ phức tạp O(ln n)
(C) Binary Searching trên Ordered Array có độ phức tạp O(log n)
[D] Searching trên Heap có độ phức tạp O(log n)

Câu 4. Trong trường hợp tốt nhất, giải thuật sort nào hiệu quả nhất:

[A] Insertion sort (B) Merge sort
(C) Quick sort (D) Selection sort

Câu 5. Số node tối thiểu của 1 **full** binary tree có depth = 3 (cây chỉ có root xem như có depth = 0) là:

[A] 7 (B) 3
(C) 5 (D) 8
(E) 15

Câu 6. Thuật ngữ nào dùng để diễn tả giải thuật O(n):

(A) Constant (B) Logarithmic
[C] Linear (D) Cubic

Câu 7. Trong trường hợp tốt nhất, giải thuật sort nào kém hiệu quả nhất:

(A) Insertion sort [B] Bubble sort
(C) Quick sort (D) Selection sort

Câu 8. Giả sử cần chuyển biểu thức trung tố sang hậu tố, cấu trúc dữ liệu nào phù hợp nhất được sử dụng để giải quyết bài toán này:

(A) Queue (B) Tree
(C) List [D] Stack

Câu 9. 1 hash table có length 10 sử dụng hash function h(k) = k mod 10, và linear probing. Sau khi chèn 6 giá trị vào 1 empty hash table, thì table chứa giá trị như sau: [_, _, 42, 23, 34, 52, 46, 33, _, _] (index của table được đánh từ 0). Các key values được chèn theo thứ tự trước sau như thế nào?

(A) 46, 42, 34, 52, 23, 33 [B] 46, 34, 42, 23, 52, 33
(C) 34, 42, 23, 52, 33, 46 (D) 42, 46, 33, 23, 34, 52

Câu 10. Con trỏ (pointers gồm front, rear) nào sẽ bị thay đổi khi insert 1 phần tử vào 1 queue đang có 1 phần tử:

(A) front (B) cả 2
[C] rear
(D) không có con trỏ nào thay đổi

Câu 11. Cho array A chứa n số nguyên có giá trị khác nhau từng đôi một. Giả sử muốn tìm giá trị nhỏ nhất trong A, trong trường hợp xấu nhất, giải pháp nào sau đây hiệu quả nhất:

(A) Sort A tăng dần, min là phần tử đầu tiên
(B) Tạo 1 BST từ A, tìm phần tử min
[C] Lặp qua cả dãy tìm phần tử min
(D) Tạo 1 MinHeap từ A, min được chứa trong root

Câu 12. Dạng nào của List phù hợp để trả lời câu hỏi "Phần tử thứ i của list chứa gì?":

[A] Lists hiện thực bằng 1 array
(B) Doubly-linked lists (danh sách liên kết 2 chiều)
(C) Singly-linked lists
(D) Singly-linked lists hoặc Doubly-linked lists đều phù hợp

Câu 13. 1 cây nhị phân khi duyệt NLR cho kết quả **14 2 1 3 11 10 7 30 40**, duyệt LRN cho kết quả **1 3 2 7 10 40 30 11 14**, duyệt LNR sẽ có kết quả là:

(A) 1 2 3 7 10 11 14 30 40
(B) 1 2 3 7 10 14 30 11 40
[C] 1 2 3 14 7 10 11 40 30
(D) 14 1 2 3 10 11 7 40 30

II. ĐIỀN KẾT QUẢ (4 điểm)

Câu 14. Cho function sau:

```
int mystery_2(Node *tree = this.root):
    if(tree == nullptr)
        return 0;
    else if (tree.getLeft() == nullptr and tree.getRight() == nullptr):
        return 1;
    return 1 + mystery_2(tree.getLeft()) + mystery_2(tree.getRight());
```

Hãy cho biết công dụng của function này:

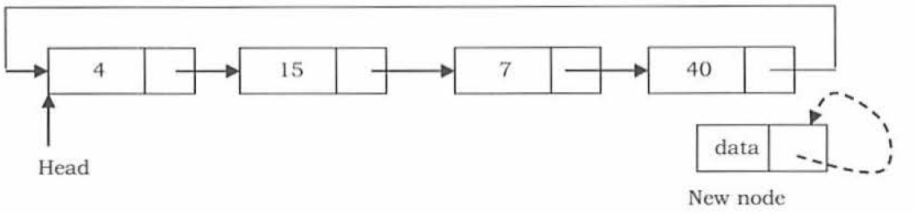
Câu 15. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], sử dụng **bubble sort** sắp xếp A **tăng dần**, sau 4 bước sắp xếp (four complete passes) thì A = ?.

Câu 16. Cho function sau:

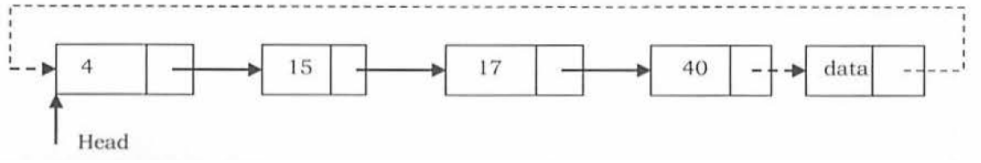
```
int mystery_1(Node *tree = this.root):
    if(tree == nullptr):
        return 0;
    elif (tree.getLeft() == nullptr and tree.getRight() == nullptr):
        return 0;
    return 1 + mystery_1(tree.getLeft()) + mystery_1(tree.getRight());
```

Hãy cho biết công dụng của function này:

Câu 17. Cho hình bên dưới:



Hãy viết các câu lệnh cần thiết để có thể chèn phần tử New node vào vị trí như hình sau, giả sử đã có 2 con trỏ Head và New node như hình:



Câu 18. Chèn các khóa 12, 18, 13, 2, 3, 23, 5 và 15 vào 1 empty hash table có size là 10, sử dụng h(k) = k mod 10 và linear probling. Hãy vẽ hash table kết quả (index từ 0).

Câu 19. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], sử dụng **insertion sort** sắp xếp A **giảm dần**, sau 4 bước sắp xếp (four complete passes) thì A = ?.

Câu 20. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], lần lượt insert các phần tử của A theo thứ tự index từ 0->7 vào 1 empty MaxHeap, hãy vẽ cây Heap cuối cùng được tạo ra.

Câu 21. Xóa lần lượt phần tử lớn nhất, lớn nhì, và lớn thứ 3 của MaxHeap ở câu trên. Hãy vẽ MaxHeap sau khi xóa.

Câu 22. Cho array A = [21, 1, 26, 45, 29, 28, 2, 9, 16, 49, 39, 27, 43, 34, 46, 40], lần lượt insert các phần tử của A theo thứ tự index từ 0->15 vào 1 cây empty AVL, hãy vẽ cây AVL cuối cùng được tạo ra.

Câu 23. Xóa lần lượt 9, và 29 của cây AVL ở câu trên. Hãy vẽ cây AVL sau khi xóa.

Câu 24. Cho array A = [21, 1, 26, 45, 29, 28, 2, 9, 16, 49, 39, 27, 43, 34, 46, 40], lần lượt insert các phần tử của A theo thứ tự index từ 0->15 vào 1 empty BST, hãy vẽ BST cuối cùng được tạo ra.

III. LẬP TRÌNH (5 điểm)

Câu 25. (1đ) (CODING STYLE)
Sinh viên sẽ có điểm câu này nếu làm ít nhất 1 câu trong các câu tiếp theo và có coding style tốt. Điểm được chấm theo 3 mức: 0, 0.5, hoặc 1.

Câu 26. Viết function **bool moveK2first(Node* head, int k)** để di chuyển node ở vị trí thứ k ra đầu danh sách. Giả sử phần tử đầu tiên nằm ở vị trí 1.

Câu 27. Viết function **int countLeave(TreeNode* root)** để đếm số lá có trong 1 cây nhị phân tổng quát.

Câu 28. Cho các khai báo sau:

```
struct StudentNode {
    int id;
    string name;
    StudentNode* next;
};

struct ClassNode {
    int id;
    string name;
    StudentNode* students;
    ClassNode* next;
};
```

Trong đó, cấu trúc StudentNode dùng để lưu trữ một node trong danh sách liên kết đơn chứa thông tin sinh viên của một lớp, ClassNode dùng để lưu trữ một node trong danh sách liên kết đơn chứa thông tin của các lớp đang quản lý trong một trường học. Thành phần students của ClassNode đại diện cho danh sách sinh viên của lớp đó.

Hãy định nghĩa hàm **bool addStudent(ClassNode* list, int classId, int studentId, string studentName)**, trong đó list là con trỏ trỏ đến phần tử đầu tiên trong danh sách lớp, classId, studentId, studentName lần lượt là mã số lớp, mã số sinh viên và tên của sinh viên. Hàm thực thực hiện chức năng sau:

- Thêm sinh viên vào đúng lớp có classId tương ứng và trả về true. Nếu không tồn tại classId đó, không thực hiện và trả về false.

- Nếu tồn tại studentId (trong toàn bộ các lớp) không thực hiện và trả về false.

Câu 29. Viết function **void printSmallerNearestNode(BSTNode* root, int key)** để tìm kiếm và in ra màn hình node duy nhất có giá trị nhỏ (hoặc bằng) hơn và gần nhất với giá trị **key** trong 1 BST (giá trị **key** vì thế có thể không tồn tại trong cây).

TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH

Họ và tên:.....
MSSV:.....

KIỂM TRA CUỐI KỲ (25/07/2020)

Môn thi: Cấu trúc dữ liệu và giải thuật Thời gian làm bài: 120 phút

☐ Không được phép dùng tài liệu ☒ Được phép dùng tài liệu giấy
Đề thi số: **9** **Đề thi gồm 6 trang.**

Chú ý: Sinh viên làm trực tiếp trên đề thi; mỗi câu hỏi trắc nghiệm chỉ chọn một đáp án đúng nhất. Các khai báo về thư viện, các hàm cơ bản và khai báo khác (nếu cần thiết) được xem như đầy đủ

I. TRẮC NGHIỆM (3 điểm)

Câu 1. Giả sử cần chuyển biểu thức trung tố sang hậu tố, cấu trúc dữ liệu nào phù hợp nhất được sử dụng để giải quyết bài toán này:
(A) Stack (B) Queue
(C) Tree (D) List

Câu 2. Trong trường hợp tốt nhất, giải thuật sort nào hiệu quả nhất:
(A) Selection sort (B) Insertion sort
(C) Merge sort (D) Quick sort

Câu 3. Con trỏ (pointers gồm front, rear) nào sẽ bị thay đổi khi insert 1 phần tử vào 1 queue đang có 1 phần tử:
(A) không có con trỏ nào thay đổi
(B) front (C) cả 2
(D) rear

Câu 4. Dạng nào của List phù hợp để trả lời câu hỏi "Phần tử thứ i của list chứa gì?":
(A) Singly-linked lists hoặc Doubly-linked lists đều phù hợp
(B) Lists hiện thực bằng 1 array
(C) Doubly-linked lists (danh sách liên kết 2 chiều)
(D) Singly-linked lists

Câu 5. 1 hash table có length 10 sử dụng hash function h(k) = k mod 10, và linear probling. Sau khi chèn 6 giá trị vào 1 empty hash table, thì table chứa giá trị như sau: [_, _, 42, 23, 34, 52, 46, 33, _, _] (index của table được đánh từ 0). Các key values được chèn theo thứ tự trước sau như thế nào?
(A) 42, 46, 33, 23, 34, 52 (B) 46, 42, 34, 52, 23, 33
(C) 46, 34, 42, 23, 52, 33 (D) 34, 42, 23, 52, 33, 46

Câu 6. Xét giải thuật dùng stack để xác định sự cân bằng của 1 chuỗi các dấu đóng, mở ngoặc. Số phần tử trong stack tại một thời điểm (AT ANY ONE TIME) lớn nhất là bao nhiêu khi phân tích chuỗi sau: (()(())(()))
(A) 1 (B) 5 hoặc nhiều hơn
(C) 3 (D) 2
(E) 4

Câu 7. Trong trường hợp tốt nhất, giải thuật sort nào kém hiệu quả nhất:
(A) Selection sort (B) Insertion sort
(C) Bubble sort (D) Quick sort

Câu 8. Xét đoạn code sau, độ phức tạp Big-O:

```
int test = 0, i, j;
for (i = 0; i < n; i++):
    for (j = 0; j < i; j++):
        test = test + i * j;
```

(A) O(1) (B) O(n)
(C) O(log n) (D) O(n^2)

Câu 9. Số node tối thiểu của 1 **full** binary tree có depth = 3 (cây chỉ có root xem như có depth = 0) là:
(A) 3 (B) 7
(C) 15 (D) 5
(E) 8

Câu 10. Chọn phát biểu sai
(A) Searching trên Heap có độ phức tạp O(log n)
(B) Linear Searching có độ phức tạp O(n)
(C) Searching trên cây AVL có độ phức tạp O(ln n)
(D) Binary Searching trên Ordered Array có độ phức tạp O(log n)

Câu 11. Thuật ngữ nào dùng để diễn tả giải thuật O(n):
(A) Cubic (B) Constant
(C) Logarithmic (D) Linear

Câu 12. Cho array A chứa n số nguyên có giá trị khác nhau từng đôi một. Giả sử muốn tìm giá trị nhỏ nhất trong A, trong trường hợp xấu nhất, giải pháp nào sau đây hiệu quả nhất:
(A) Tạo 1 MinHeap từ A, min được chứa trong root
(B) Sort A tăng dần, min là phần tử đầu tiên
(C) Tạo 1 BST từ A, tìm phần tử min
(D) Lặp qua cả dãy tìm phần tử min

Câu 13. 1 cây nhị phân khi duyệt NLR cho kết quả **14 2 1 3 11 10 7 30 40**, duyệt LRN cho kết quả **1 3 2 7 10 40 30 11 14**, duyệt LNR sẽ có kết quả là:
(A) 14 1 2 3 10 11 7 40 30 (B) 1 2 3 7 10 11 14 30 40
(C) 1 2 3 7 10 14 30 11 40 (D) 1 2 3 14 7 10 11 40 30

II. ĐIỀN KẾT QUẢ (4 điểm)

Câu 14. Cho function sau:

```
int mystery_1(Node *tree = this.root):
    if(tree == nullptr):
        return 0;
    elif (tree.getLeft() == nullptr and tree.getRight() == nullptr):
        return 0;
    return 1 + mystery_1(tree.getLeft()) + mystery_1(tree.getRight());
```

Hãy cho biết công dụng của function này:

Câu 15. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], sử dụng **insertion sort** sắp xếp A **giảm dần**, sau 4 bước sắp xếp (four complete passes) thì A = ?.

Câu 16. Cho function sau:

```
int mystery_2(Node *tree = this.root):
    if(tree == nullptr)
        return 0;
    else if (tree.getLeft() == nullptr and tree.getRight() == nullptr):
        return 1;
    return 1 + mystery_2(tree.getLeft()) + mystery_2(tree.getRight());
```

Hãy cho biết công dụng của function này:

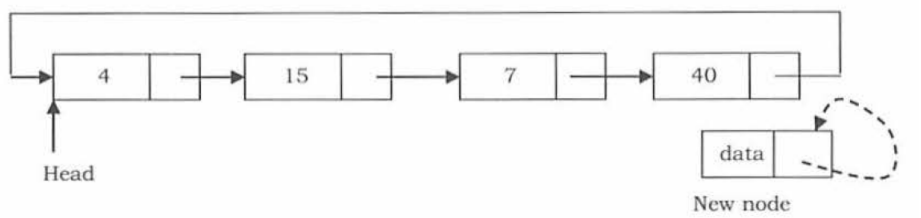
Câu 17. Chèn các khóa 12, 18, 13, 2, 3, 23, 5 và 15 vào 1 empty hash table có size là 10, sử dụng h(k) = k mod 10 và linear probling. Hãy vẽ hash table kết quả (index từ 0).

Câu 18. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], sử dụng **bubble sort** sắp xếp A **tăng dần**, sau 4 bước sắp xếp (four complete passes) thì A = ?.

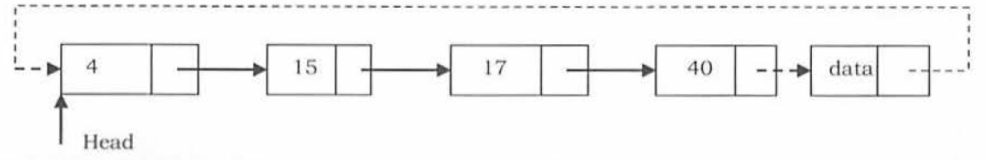
Câu 19. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], lần lượt insert các phần tử của A theo thứ tự index từ 0->7 vào 1 empty MaxHeap, hãy vẽ cây Heap cuối cùng được tạo ra.

Câu 20. Xóa lần lượt phần tử lớn nhất, lớn nhì, và lớn thứ 3 của MaxHeap ở câu trên. Hãy vẽ MaxHeap sau khi xóa.

Câu 21. Cho hình bên dưới:



Hãy viết các câu lệnh cần thiết để có thể chèn phần tử New node vào vị trí như hình sau, giả sử đã có 2 con trỏ Head và New node như hình:



Câu 22. Cho array A = [21, 1, 26, 45, 29, 28, 2, 9, 16, 49, 39, 27, 43, 34, 46, 40], lần lượt insert các phần tử của A theo thứ tự index từ 0->15 vào 1 cây empty AVL, hãy vẽ cây AVL cuối cùng được tạo ra.

Câu 23. Xóa lần lượt 9, và 29 của cây AVL ở câu trên. Hãy vẽ cây AVL sau khi xóa.

Câu 24. Cho array A = [21, 1, 26, 45, 29, 28, 2, 9, 16, 49, 39, 27, 43, 34, 46, 40], lần lượt insert các phần tử của A theo thứ tự index từ 0->15 vào 1 empty BST, hãy vẽ BST cuối cùng được tạo ra.

III. LẬP TRÌNH (5 điểm)

Câu 25. (1đ) (CODING STYLE)
Sinh viên sẽ có điểm câu này nếu làm ít nhất 1 câu trong các câu tiếp theo và có coding style tốt. Điểm được chấm theo 3 mức: 0, 0.5, hoặc 1.

Câu 26. Viết function **bool moveK2first(Node* head, int k)** để di chuyển node ở vị trí thứ k ra đầu danh sách. Giả sử phần tử đầu tiên nằm ở vị trí 1.

Câu 27. Viết function **int countLeave(TreeNode* root)** để đếm số lá có trong 1 cây nhị phân tổng quát.

Câu 28. Cho các khai báo sau:

```
struct StudentNode {
    int id;
    string name;
    StudentNode* next;
};

struct ClassNode {
    int id;
    string name;
    StudentNode* students;
    ClassNode* next;
};
```

Trong đó, cấu trúc StudentNode dùng để lưu trữ một node trong danh sách liên kết đơn chứa thông tin sinh viên của một lớp, ClassNode dùng để lưu trữ một node trong danh sách liên kết đơn chứa thông tin của các lớp đang quản lý trong một trường học. Thành phần students của ClassNode đại diện cho danh sách sinh viên của lớp đó.

Hãy định nghĩa hàm **bool addStudent(ClassNode* list, int classId, int studentId, string studentName)**, trong đó list là con trỏ trỏ đến phần tử đầu tiên trong danh sách lớp, classId, studentId, studentName lần lượt là mã số lớp, mã số sinh viên và tên của sinh viên. Hàm thực thực hiện chức năng sau:

- Thêm sinh viên vào đúng lớp có classId tương ứng và trả về true. Nếu không tồn tại classId đó, không thực hiện và trả về false.

- Nếu tồn tại studentId (trong toàn bộ các lớp) không thực hiện và trả về false.

Câu 29. Viết function **void printSmallerNearestNode(BSTNode* root, int key)** để tìm kiếm và in ra màn hình node duy nhất có giá trị nhỏ (hoặc bằng) hơn và gần nhất với giá trị **key** trong 1 BST (giá trị **key** vì thế có thể không tồn tại trong cây).

Chủ nhiệm Khoa/Bộ môn	Giảng viên ra đề
	ThS. Vương Bá Thịnh

TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH

KIỂM TRA CUỐI KỲ (25/07/2020)

Môn thi: Cấu trúc dữ liệu và giải thuật Thời gian làm bài: 120 phút

☐ Không được phép dùng tài liệu ☒ Được phép dùng tài liệu giấy
Đề thi số: 9

I. TRẮC NGHIỆM (3 điểm)

Câu 1. Giả sử cần chuyển biểu thức trung tố sang hậu tố, cấu trúc dữ liệu nào phù hợp nhất được sử dụng để giải quyết bài toán này:

A Stack
(B) Queue
(C) Tree
(D) List

Câu 2. Trong trường hợp tốt nhất, giải thuật sort nào hiệu quả nhất:

(A) Selection sort
B Insertion sort
(C) Merge sort
(D) Quick sort

Câu 3. Con trỏ (pointers gồm front, rear) nào sẽ bị thay đổi khi insert 1 phần tử vào 1 queue đang có 1 phần tử:

(A) không có con trỏ nào thay đổi
(B) front
(C) cả 2
D rear

Câu 4. Dạng nào của List phù hợp để trả lời câu hỏi "Phần tử thứ i của list chứa gì?":

(A) Singly-linked lists hoặc Doubly-linked lists đều phù hợp
B Lists hiện thực bằng 1 array
(C) Doubly-linked lists (danh sách liên kết 2 chiều)
(D) Singly-linked lists

Câu 5. 1 hash table có length 10 sử dụng hash function h(k) = k mod 10, và linear probling. Sau khi chèn 6 giá trị vào 1 empty hash table, thì table chứa giá trị như sau: [_, _, 42, 23, 34, 52, 46, 33, _, _] (index của table được đánh từ 0). Các key values được chèn theo thứ tự trước sau như thế nào?

(A) 42, 46, 33, 23, 34, 52
(B) 46, 42, 34, 52, 23, 33
C 46, 34, 42, 23, 52, 33
(D) 34, 42, 23, 52, 33, 46

Câu 6. Xét giải thuật dùng stack để xác định sự cân bằng của 1 chuỗi các dấu đóng, mở ngoặc. Số phần tử trong stack tại một thời điểm (AT ANY ONE TIME) lớn nhất là bao nhiêu khi phân tích chuỗi sau: (()(())(()))

(A) 1
B 5 hoặc nhiều hơn
(C) 3
(D) 2
(E) 4

Câu 7. Trong trường hợp tốt nhất, giải thuật sort nào kém hiệu quả nhất:

(A) Selection sort
(B) Insertion sort
C Bubble sort
(D) Quick sort

Câu 8. Xét đoạn code sau, độ phức tạp Big-O:

```
int test = 0, i, j;
for (i = 0; i < n; i++):
    for (j = 0; j < i; j++):
        test = test + i * j;
```

(A) O(1)
(B) O(n)
(C) O(log n)
D O(n^2)

Câu 9. Số node tối thiểu của 1 **full** binary tree có depth = 3 (cây chỉ có root xem như có depth = 0) là:

(A) 3
B 7
(C) 15
(D) 5
(E) 8

Câu 10. Chọn phát biểu sai

A Searching trên Heap có độ phức tạp O(log n)
(B) Linear Searching có độ phức tạp O(n)
(C) Searching trên cây AVL có độ phức tạp O(ln n)
(D) Binary Searching trên Ordered Array có độ phức tạp O(log n)

Câu 11. Thuật ngữ nào dùng để diễn tả giải thuật O(n):

(A) Cubic
(B) Constant
(C) Logarithmic
D Linear

Câu 12. Cho array A chứa n số nguyên có giá trị khác nhau từng đôi một. Giả sử muốn tìm giá trị nhỏ nhất trong A, trong trường hợp xấu nhất, giải pháp nào sau đây hiệu quả nhất:

(A) Tạo 1 MinHeap từ A, min được chứa trong root
(B) Sort A tăng dần, min là phần tử đầu tiên
(C) Tạo 1 BST từ A, tìm phần tử min
D Lặp qua cả dãy tìm phần tử min

Câu 13. 1 cây nhị phân khi duyệt NLR cho kết quả **14 2 1 3 11 10 7 30 40**, duyệt LRN cho kết quả **1 3 2 7 10 40 30 11 14**, duyệt LNR sẽ có kết quả là:

(A) 14 1 2 3 10 11 7 40 30
(B) 1 2 3 7 10 11 14 30 40
(C) 1 2 3 7 10 14 30 11 40
D 1 2 3 14 7 10 11 40 30

II. ĐIỀN KẾT QUẢ (4 điểm)

Câu 14. Cho function sau:

```
int mystery_1(Node *tree = this.root):
    if(tree == nullptr):
        return 0;
    elif (tree.getLeft() == nullptr and tree.getRight() == nullptr):
        return 0;
    return 1 + mystery_1(tree.getLeft()) + mystery_1(tree.getRight());
```

Hãy cho biết công dụng của function này:

Câu 15. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], sử dụng **insertion sort** sắp xếp A **giảm dần**, sau 4 bước sắp xếp (four complete passes) thì A = ?.

Câu 16. Cho function sau:

```
int mystery_2(Node *tree = this.root):
    if(tree == nullptr)
        return 0;
    else if (tree.getLeft() == nullptr and tree.getRight() == nullptr):
        return 1;
    return 1 + mystery_2(tree.getLeft()) + mystery_2(tree.getRight());
```

Hãy cho biết công dụng của function này:

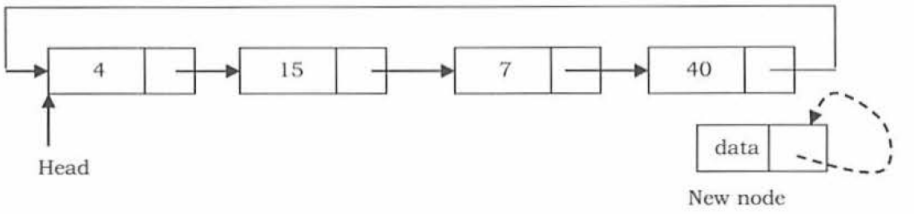
Câu 17. Chèn các khóa 12, 18, 13, 2, 3, 23, 5 và 15 vào 1 empty hash table có size là 10, sử dụng h(k) = k mod 10 và linear probling. Hãy vẽ hash table kết quả (index từ 0).

Câu 18. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], sử dụng **bubble sort** sắp xếp A **tăng dần**, sau 4 bước sắp xếp (four complete passes) thì A = ?.

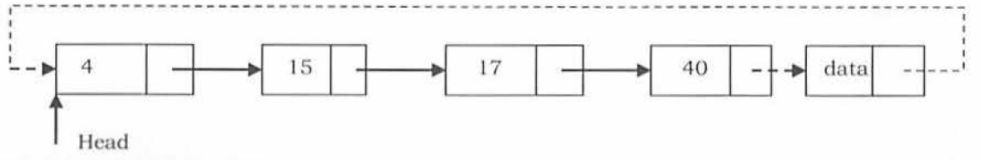
Câu 19. Cho array A = [54, 26, 93, 17, 77, 31, 44, 55, 20], lần lượt insert các phần tử của A theo thứ tự index từ 0->7 vào 1 empty MaxHeap, hãy vẽ cây Heap cuối cùng được tạo ra.

Câu 20. Xóa lần lượt phần tử lớn nhất, lớn nhì, và lớn thứ 3 của MaxHeap ở câu trên. Hãy vẽ MaxHeap sau khi xóa.

Câu 21. Cho hình bên dưới:



Hãy viết các câu lệnh cần thiết để có thể chèn phần tử New node vào vị trí như hình sau, giả sử đã có 2 con trỏ Head và New node như hình:



Câu 22. Cho array A = [21, 1, 26, 45, 29, 28, 2, 9, 16, 49, 39, 27, 43, 34, 46, 40], lần lượt insert các phần tử của A theo thứ tự index từ 0->15 vào 1 cây empty AVL, hãy vẽ cây AVL cuối cùng được tạo ra.

Câu 23. Xóa lần lượt 9, và 29 của cây AVL ở câu trên. Hãy vẽ cây AVL sau khi xóa.

Câu 24. Cho array A = [21, 1, 26, 45, 29, 28, 2, 9, 16, 49, 39, 27, 43, 34, 46, 40], lần lượt insert các phần tử của A theo thứ tự index từ 0->15 vào 1 empty BST, hãy vẽ BST cuối cùng được tạo ra.

III. LẬP TRÌNH (5 điểm)

Câu 25. (1đ) (CODING STYLE)
Sinh viên sẽ có điểm câu này nếu làm ít nhất 1 câu trong các câu tiếp theo và có coding style tốt. Điểm được chấm theo 3 mức: 0, 0.5, hoặc 1.

Câu 26. Viết function **bool moveK2first(Node* head, int k)** để di chuyển node ở vị trí thứ k ra đầu danh sách. Giả sử phần tử đầu tiên nằm ở vị trí 1.

Câu 27. Viết function **int countLeave(TreeNode* root)** để đếm số lá có trong 1 cây nhị phân tổng quát.

Câu 28. Cho các khai báo sau:

```
struct StudentNode {
    int id;
    string name;
    StudentNode* next;
};

struct ClassNode {
    int id;
    string name;
    StudentNode* students;
    ClassNode* next;
};
```

Trong đó, cấu trúc StudentNode dùng để lưu trữ một node trong danh sách liên kết đơn chứa thông tin sinh viên của một lớp, ClassNode dùng để lưu trữ một node trong danh sách liên kết đơn chứa thông tin của các lớp đang quản lý trong một trường học. Thành phần students của ClassNode đại diện cho danh sách sinh viên của lớp đó.

Hãy định nghĩa hàm **bool addStudent(ClassNode* list, int classId, int studentId, string studentName)**, trong đó list là con trỏ trỏ đến phần tử đầu tiên trong danh sách lớp, classId, studentId, studentName lần lượt là mã số lớp, mã số sinh viên và tên của sinh viên. Hàm thực thực hiện chức năng sau:

- Thêm sinh viên vào đúng lớp có classId tương ứng và trả về true. Nếu không tồn tại classId đó, không thực hiện và trả về false.

- Nếu tồn tại studentId (trong toàn bộ các lớp) không thực hiện và trả về false.

Câu 29. Viết function **void printSmallerNearestNode(BSTNode* root, int key)** để tìm kiếm và in ra màn hình node duy nhất có giá trị nhỏ (hoặc bằng) hơn và gần nhất với giá trị **key** trong 1 BST (giá trị **key** vì thế có thể không tồn tại trong cây).