KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

BỘ MÔN KHOA HỌC MÁY TÍNH

Môn thi: Cấu trúc dữ liệu và giải thuật

Họ và tên: .....

MSSV: .....

ĐỀ THI GIỮA KÌ

MÔN THI: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

THỜI GIAN: 75 PHÚT

☒ Sinh viên được phép sử dụng tài liệu ☐ Sinh viên không được phép sử dụng tài liệu

MÃ ĐỀ: 1001

Hướng dẫn làm bài:

Sinh viên phải ghi họ tên và mã số sinh viên vào đề thi. ***Sinh viên phải nộp lại đề thi***
Phần I: TRẮC NGHIỆM. Sinh viên làm trên đề thi bằng cách khoanh tròn câu trả lời đúng nhất.
Phần II: ĐIỀN CHỖ TRỐNG. Sinh viên làm trên đề thi
Phần III: VIẾT CHƯƠNG TRÌNH. Sinh viên làm trên giấy thi

PHẦN I: TRẮC NGHIỆM (3.0 điểm) *Để được trọn số điểm này, sinh viên chỉ cần trả lời đúng 10 câu*

1. Độ phức tạp thời gian của giải thuật được xác định theo tiêu chí nào dưới đây:

A. Thời gian thực thi của giải thuật tính ra μs (micro giây)
B. Số lượng câu lệnh
C. Số lượng những thao tác tốn kém thời gian (key operations)
D. Đáp án A, B, C đều sai

2. Biết rằng Statement A và Statement B đều có độ phức tạp là O(1). Hãy cho biết độ phức tạp của giải thuật sau:

```
for(int i = 0; i< n; i++)
    for(int  j = 0; j< i; j++)
        Stament A;
for(int k = 0; k<n; k*=2)
    Statement B;
```

A. $n^2 \log_2 n$ B. $n^2 + \log_2 n$ C. n^2 D. Đáp án A, B, C đều sai

3. Nhận định nào sau đây là SAI:

A. Trong hàm đệ quy có thể có nhiều hơn một điểm dừng (base case)
B. Nếu hàm đệ quy có điểm dừng (base case) thì sẽ không xảy ra đệ quy vô hạn khi gọi hàm
C. Nếu lời gọi hàm đệ quy không đi đến điểm dừng (base case) thì đệ quy vô hạn sẽ xuất hiện
D. A và C là đúng

4. Cho foo(...) là một hàm đệ quy, x là biến toàn cục:

```
int x = 0;
int foo(int n){
    if(n <=0)     return 0;
    else {
```

```
        cin >> x;
        return foo(n - 1) + x;
    }
}
```

Hãy cho biết lời gọi hàm foo(4) trả về giá trị bao nhiêu, nếu các giá trị nhập vào lần lượt là 4, 9, 6 và 2.

A. 21 B. 0 C. 8 D. 9

5. Giả sử head và tail lần lượt trỏ vào phần tử đầu và phần tử cuối của danh sách liên kết, khi thực hiện thao tác chèn phần tử vào danh sách, thì

A. Chỉ cần cập nhật con trỏ head
B. Chỉ cần cập nhật con trỏ tail
C. Phải cập nhật cả head lẫn tail
D. Có thể phải cập nhật cả head lẫn tail

6. Danh sách liên kết KHÔNG có đặc điểm nào dưới đây:

A. Khi thực hiện thao tác chèn hoặc xóa, không cần di chuyển các phần tử
B. Không cần phải xin cấp phát bộ nhớ trước
C. Tốc độ truy xuất các phần tử giống như nhau
D. Lượng bộ nhớ cấp phát cho danh sách liên kết tỷ lệ thuận với chiều dài của danh sách

7. Xét cây nhị phân, trong trường hợp cây như thế nào thì kết quả duyệt tiền thứ tự và trung thứ tự giống nhau:

A. Nút gốc không có con bên trái
B. Nút gốc không có con bên phải
C. Tất cả các nút trong chỉ có con bên trái
D. Tất cả các nút trong chỉ có con bên phải

8. Giả sử ta dùng mảng V[1..n] để hiện thực ngăn xếp, khi ngăn xếp rỗng, vị trí đỉnh ngăn xếp top có giá trị là n+1, khi thêm x vào ngăn xếp thì thao tác nào dưới đây là đúng

A. top:=top+1; V [top]:=x
B. V [top]:=x; top:=top+1
C. top:=top-1; V [top]:=x
D. V [top]:=x; top:=top-1

9. Đặc điểm chung của ngăn xếp và hàng đợi là:

A. Đều là cấu trúc dữ liệu FIFO
B. Đều là cấu trúc dữ liệu LIFO
C. Thao tác chèn và xóa đều được thực hiện ở đầu mút
D. Hai cấu trúc dữ liệu này không có điểm chung nào

10. Nhận định nào sau đây về cây nhị phân tìm kiếm là đúng

A. Trường hợp tìm kiếm tốt nhất có độ phức tạp là $O(\log_2 n)$
B. Trong trường hợp tốt nhất, xây dựng cây nhị phân tìm kiếm n nút sẽ có độ phức tạp là $O(\log_2 n)$
C. Cả 2 đáp án A và B đều đúng
D. Cả 2 đáp án A và B đều sai

11. Chiều cao H của cây nhị phân có 1025 nút có giá trị

A. 11 B. 10 C. từ 11 đến 1025 D. từ 10 đến 1024

12. Nhận định nào dưới đây đúng

A. Khi kiểu của con trỏ và kiểu của biến không giống nhau, có thể sử dụng ép kiểu tường minh để chuyển kiểu của biến thành kiểu của con trỏ. Ví dụ: int* p; float a; p = &((int) a);

```
        cin >> x;
        return foo(n - 1) + x;
    }
}
```

Hãy cho biết

B. Con trỏ thực chất cũng là biến, cũng cần không gian trong bộ nhớ. Nó chiếm một lượng bộ nhớ đúng bằng lượng bộ nhớ mà nó trỏ đến.
C. Con trỏ chiếm một lượng bộ nhớ cố định, không phụ thuộc vào kiểu của nó
D. Lượng bộ nhớ mà con trỏ chiếm phụ thuộc vào không gian trống của bộ nhớ.

PHẦN II: ĐIỀN CHỖ TRỐNG (3.5 điểm) *Để được trọn số điểm này, sinh viên chỉ cần trả lời đúng 10 câu*

1. **Độ phức tạp O(n) của giải thuật sau là** _____

```
x = n; // n là số nguyên lớn
while (x>=(y+1)*(y+1))
    y++;
```

2. **Độ phức tạp O(n) của giải thuật sau là** _____

```
i=1; j=0;
while(i+j<=n)  //n là số nguyên lớn
    if (i>j) j++;
    else     i++;
```

3. **Cho hàm**

```
void foo1(int w) {
    if(w > 0) {
        foo1(w - 1);
        cout << w << "   ";
        foo1(w - 1);
    }
}
```

Lời gọi hàm foo1(3) sẽ xuất ra màn hình giá trị _____

4. **Cho hàm đệ quy**

```
int  foo2(int n) {
    if(n <= 3)    return 1;
    else return foo2(n - 1) + foo2(n-2) + foo2(n - 3);
}
```

Lời gọi hàm foo2(10) trả về kết quả bằng _____

5. **Giá trị của biểu thức tiền tố:** + * * 9 + 3 18 + * 3 9 7 8, **bằng** _____

6. **Dùng mảng có kích thước bằng 4 để hiện thực hàng đợi vòng, giả sử hiện tại giá trị của rear và front lần lượt là 0 và 3, bây giờ xóa một phần tử khỏi hàng đợi, rồi thêm vào hàng đợi hai phần tử nữa, lúc đó giá trị của rear và front lần lượt là bao nhiêu:** _____

7. **Dạng biểu diễn hậu tố (postfix) của biểu thức:** 8 – (3 + 5) * (5 – 6 / 2) là

_____

8. **Sử dụng ngăn xếp để hiện thực giải thuật kiểm tra các dấu ngoặc trong một biểu thức có đi cặp với nhau không (Check the brackets are correctly matched or not). Khi kiểm tra biểu thức (0(0)(0)), thì số lượng dấu ngoặc nhiều nhất trong ngăn xếp tại một thời điểm nào đó bằng**

_____

9. Dùng mảng A[0 .. m-1] để hiện thực hàng đợi vòng (circular queue), khi thêm phần tử vào hàng đợi, thì giá trị rear phải cập nhật như sau: _____

10. Cho cây nhị phân có các nút A, B, C, D, E, F, G, H, I, J. Duyệt tiền thứ tự (PreOrder) cây nhị phân, ta có ABECDFGHIJ. Duyệt trung thứ tự (InOrder), ta có EBCDAFHIGJ. Duyệt hậu thứ tự, ta sẽ có kết quả là _____

11. Từ 4 nút có thể tạo được _____ cây nhị phân có hình dạng khác nhau

12. Cho đoạn chương trình sau:

```
void func(int* a, int* &b){   int *t; t = a; a = b;b = t;  }
void main() {
        int *p1 = new int; *p1 = 20;
        int *p2 = new int; *p2 = 10;
        func(p1, p2);
        printf("%d, %d", *p1, *p2);
}
```

Kết xuất ra màn hình của đoạn chương trình trên là: _____

PHẦN III: VIẾT CHƯƠNG TRÌNH

1. **Phép hợp hai tập hợp (2.5 điểm)**

Dùng danh sách liên kết để biểu diễn **tập hợp** với phần tử là các số nguyên. Nút trong một danh sách liên kết được định nghĩa như sau:

```
struct node { int data; node *next; };
```

Giả sử **ha** và **hb** là 2 con trỏ lần lượt trỏ vào đầu 2 danh sách liên kết A và B có các phần tử là số nguyên và được sắp xếp theo thứ tự tăng dần. Hãy viết hàm ListUnion trả về kết quả của **phép hợp** hai tập hợp. Yêu cầu:

- Prototype của hàm là: **node* ListUnion(node* ha, node* hb)**
- Sinh viên không được sử dụng **new** để tạo ra các nút trong danh sách kết quả mà lợi dụng các nút sẵn có trong hai danh sách A và B.
- Dùng toán tử **delete** để thu hồi bộ nhớ đối với các nút còn thừa (nếu có) trong hai danh sách A, B.
- Sinh viên có thể dùng mã giả hoặc ngôn ngữ C++

2. **Phán đoán cây có phải là cây nhị phân đầy đủ hay không (1.0 điểm)**

Nút trong một cây nhị phân được định nghĩa như sau:

```
struct node{ int data;    node* left;  node* right; };
```

Hãy viết hàm : **int JudgeComplete(node* bt)** cho biết cây có phải là cây nhị phân đầy đủ hay không. Hàm trả về 1, nếu cây là cây nhị phân đầy đủ. Ngược lại, hàm trả về 0. Sinh viên có thể dùng mã giả hoặc ngôn ngữ C++. Sinh viên được sử dụng các cấu trúc dữ liệu khác như danh sách, ngăn xếp, hàng đợi.

----------HẾT----------

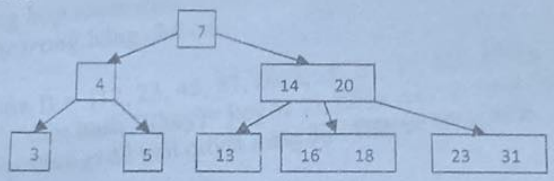
ĐÁP ÁN ĐỀ THI: [redacted] 1004

PHẦN TRẮC NGHIỆM

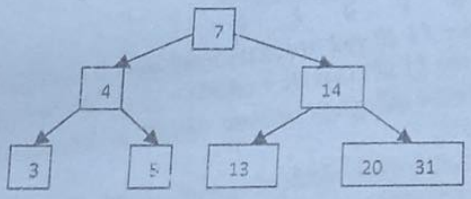
1	2	3	4	5	6	7	8	9	10	11	12
B	C	D	C	B	D	B	C	A	B	C	D

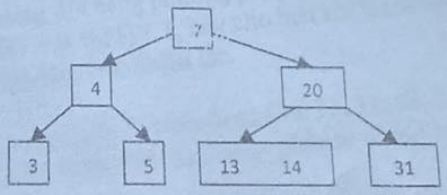
PHẦN ĐIỀN VÀO CHỖ TRỐNG

1. Thiết kế hàm hash, giải quyết xung đột
2. (21, 32, 35, 48, 37, 56, 64, 92, 85, 63, 71)
3. (9, 21, 10, 32, 37, 35, 64, 48, 85, 63, 71, 92, 56)
4. Cây B –Tree

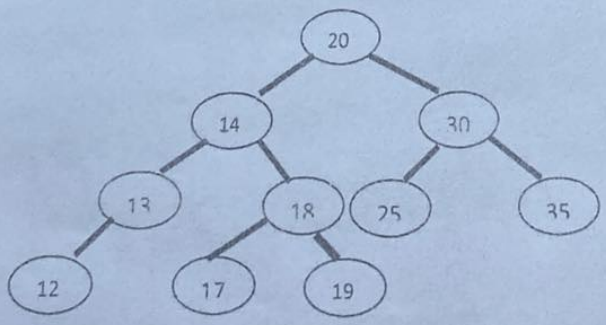


5. Cây B-Tree , xóa 16, 18, 23





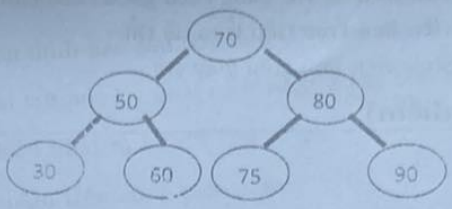
6. Cây AVL



7. Bảng hash

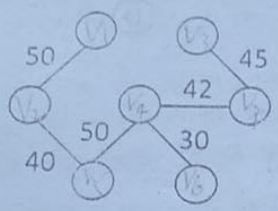
	56	23	79		38		62	41	18	
0	1	2	3	4	5	6	7	8	9	10

8. Cây AVL



9. 3, 2, 1, 1

10. Cây phủ tối thiểu



11. Duyệt đồ thị

a) 0, 4, 3, 8, 9, 7, 6, 5, 2, 1
b) 1, 0, 2, 7, 4, 8, 6, 3, 5, 9

12. Sắp xếp

46	74	53	14	26	38	86	65	27	34
14	46	74	53	26	27	38	86	65	34
14	26	46	74	53	27	34	38	86	65
14	26	27	46	74	53	34	38	65	86
14	26	27	34	46	74	53	38	65	86
14	26	27	34	38	46	74	53	65	86
14	26	27	34	38	46	53	74	65	86
14	26	27	34	38	46	53	65	74	86