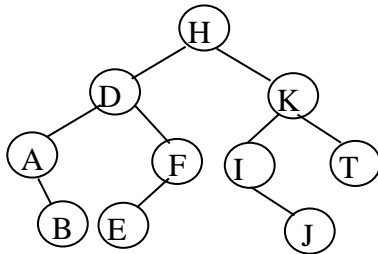


Giải đề trắc nghiệm Cấu Trúc Dữ Liệu

(Đề cô Tuyển HK1 năm 2020)

1. Trong danh sách đặc, khi xóa phần tử tại vị trí p trong danh sách ta cần phải:
 - a. Luôn luôn dịch chuyển các phần tử từ p đến L.Last ra sau một vị trí.
 - b. Có thể dịch chuyển các phần tử từ p đến L.Last ra trước một vị trí.
 - c. Luôn luôn dịch chuyển các phần tử từ p+1 đến L.Last ra sau một vị trí.
 - d. **Có thể dịch chuyển các phần tử từ p+1 đến L.Last ra trước một vị trí. (Xóa xuôi chèn ngược, chuyển dịch 1 vị trí)**
2. Cho cây nhị phân sau:

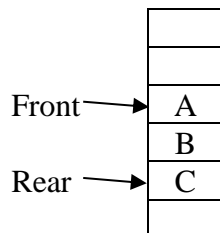


Cây nhị phân đã cho là cây tìm kiếm nhị phân
Đúng (Do các node trái < root < node phải)
 Sai

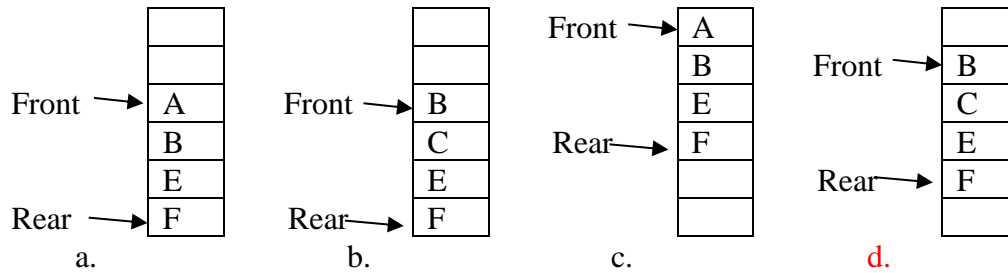
3. Danh sách duyệt NLR của cây nhị phân ở câu 2 là:
 - a. A, B, D, E, F, H, I, J, K, T
 - b. H, D, A, B, E, F, K, I, J, T
 - c. **H, D, A, B, F, E, K, I, J, T** (root ở đầu và loại trừ)
 - d. B, A, E, F, D, J, I, T, K, H
4. Danh sách duyệt LRN của cây nhị phân ở câu 2 là:
 - a. A, B, D, E, F, H, I, J, K, T
 - b. H, D, A, B, E, F, K, I, J, T
 - c. H, D, A, B, F, E, K, I, J, T
 - d. **B, A, E, F, D, J, I, T, K, H** (root ở cuối)
5. Danh sách duyệt trung tự **tổng quát** của cây nhị phân ở câu 2 là: (Phân biệt tổng quát và LNR: Khác chỗ nếu nút đó chỉ có 1 node con thì LNR sẽ phân biệt là node Left hoặc node Right, còn duyệt Trung tự thì coi như nhau)
 - a. **B, A, D, E, F, H, J, I, K, T**
 - b. B, A, D, E, F, H, I, J, K, T
 - c. A, B, D, E, F, H, I, J, K, T
 - d. A, B, D, E, F, H, J, I, K, T
6. Giá trị biểu thức tiền tố /, *, -, 2, 1, +, 3, 4, *, 7, -, 6, 5 là:
 - a. -7
 - b. -1
 - c. **1**
 - d. 7

Hướng dẫn: Duyệt tiền tố sẽ theo dạng NLR => (/) là Root, (5) là node cuối bên phải
 Cây đồ thị có lá phải là số, nên từ Root (/) sẽ đi xuống, gặp Leaf (số) thì dừng => Có 1 nhánh chính là: (/) => (*) => (-) => (2) (Tự vẽ thêm)
 Cuối cùng còn lại: 1*7/7*1 = 1 => B
7. Khi thêm phần tử đầu tiên vào danh sách liên kết L rỗng có ô đầu mục (Có Node Header) thì:
 - a. L luôn luôn thay đổi
 - b. L có thể thay đổi

- c. **L luôn luôn không đổi** (Có thể Đáp án nói đến cấu trúc của dklk không thay đổi
- Trước sau khi thêm, node Header đều đứng ở đầu)
8. Vị trí để thêm phần tử vào hàng(hàng đợi - Queue) là
- Vị trí đầu hàng
 - Vị trí cuối hàng**
 - Vị trí bất kỳ trong hàng
9. Khi xóa phần tử ra khỏi danh sách đặc L không rỗng thì ta phải luôn luôn thực hiện câu lệnh:
- L.Last--;
 - L->Last--;**
 - L->Last++;
 - Tất cả đều sai
10. Giá trị biểu thức hậu tố 1, 2, -, 3, 4, +, *, 7, 6, 5, -, *, / là:
- 1**
 - 1
 - 7
 - Một giá trị khác
- Hướng dẫn: Biểu thức duyệt hậu tố là tiền đề cho thao tác trên Stack nên chỉ cần tính từ từ, từ Trái sang phải, gặp dấu thì xử lý hai thành trước dấu đó:
- 1-2, 3,... = -1, 3, 4, +...
 - 1, 3+4,... = -1, 7, *,...
 - 1*7, 7,... = -7, 7, 6, 5, -,...
 - 7, 7, 6-5, *,... = -7, 7, 1, *,...
 - 7, 7*1, / = -7, 7, / = -7 / 7 = -1 => A
11. Lấy nội dung phần tử tại vị trí p trong danh sách liên kết kép L không có ô đầu mục (Không có Header) ta thực hiện lệnh:
- return L->Element;
 - return L->Next->Element;
 - return p->Element;**
 - return p->Next->Element;
12. Sau khi thêm phần tử X vào vị trí p trong danh sách đặc L, ta luôn luôn có mệnh đề sau là đúng:
- L.Elements[p-1]==X;
 - L.Elements[p]==X;
 - L.Elements[p+1]==X;
 - Tất cả đều sai.** (Vị trí = Chỉ số +1, có khi câu A sai chỗ có hai dấu bằng :v)
13. Khi thêm phần tử X vào bảng băm mở, ta sẽ xen phần tử đó vào đầu của danh sách ở vị trí:
- Bucket 0.
 - Bucket B-1
 - Bucket bất kỳ
 - Bucket H(X) (Băm lần 1)**
14. Cho hàng cài đặt bằng mảng tĩnh tiến như sau:



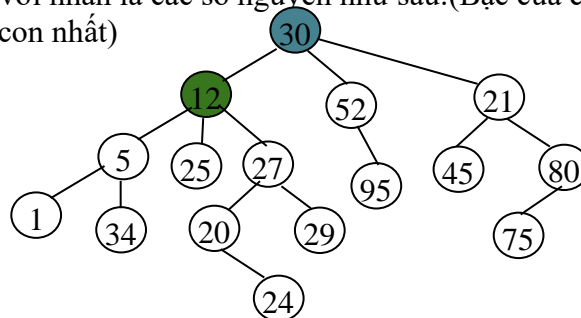
Hình ảnh hàng sau khi thêm phần tử E, F và xóa đi 1 phần tử là:



Giải:

- 1 Thêm E $\Rightarrow ++Rear = MaxLength \ \&\& \ !FullQueue \Rightarrow$ Dịch tất cả lên (Front - 1) đơn vị \Rightarrow Thứ tự gồm A, B, C, E, [], []
 - 2 Thêm F vào \Rightarrow Thứ tự gồm A, B, C, E, F, []
 - 3 Xóa 1 phần tử, Queue xóa phần tử ở Front \Rightarrow Xóa A, Front++
- Thứ Tự đúng như đáp án D

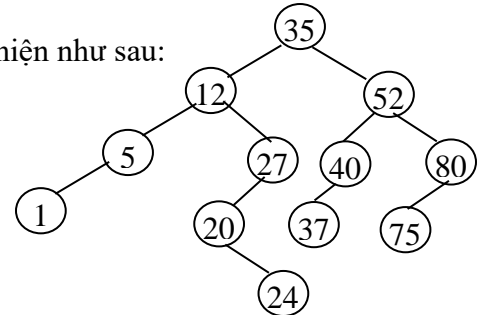
15. Khi thêm phần tử vào danh sách đặc không đầy, ta luôn luôn phải thực hiện lệnh
 - a. L.Last++;
 - b. **L->Last ++;**
 - c. L->Last--;
 - d. Tất cả đều sai
16. Khi xóa phần tử ra khỏi hàng cài đặt bằng mảng vòng thì
 - a. Front luôn luôn thay đổi và Rear luôn luôn không đổi
 - b. Front luôn luôn thay đổi và Rear có thể thay đổi
 - c. **Front có thể thay đổi và Rear có thể thay đổi** (Trường hợp trong Queue chỉ còn 1 phần tử thì Front = Rear = -1)
 - d. Front có thể thay đổi và Rear luôn luôn không đổi
17. Vị trí phần tử cần xóa ra khỏi danh sách đặc trong phép toán Delete_List (p,L) chỉ hợp lệ khi:
 - a. $0 \leq p \leq L.Last$
 - b. $0 < p \leq L.Last+1$
 - c. $1 \leq p \leq L.Last+1$
 - d. **$1 \leq p < L.Last+1$** (Vị trí = Chỉ số +1 \Rightarrow mà chỉ số mảng từ 0...(L.Last-1) nên p chỉ cần cộng thêm 1 vào 2 vế)
18. Cho cây tổng quát với nhãn là các số nguyên như sau: (Bậc của cây là số node con của node có nhiều con nhất)



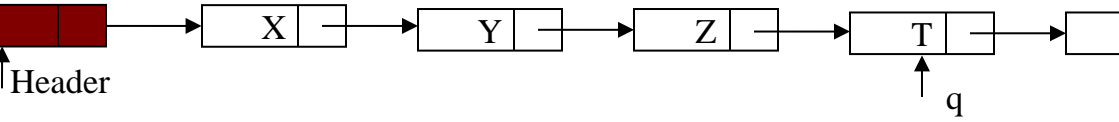
Bậc của cây là: (Nhìn vào thấy 2 node nhiều con nhất là 30 và 12)

- a. 1
 - b. 2
 - c. **3**
 - d. 4
19. Danh sách duyệt theo mức của cây ở câu 18 là: (Duyệt theo mức là duyệt từng hàng từ trên xuống từ Trái sang Phải)

- a. 24, 1, 34, 20, 29, 75, 5, 25, 27, 95, 45, 80, 12, 52, 21, 30
 - b. 1, 34, 5, 25, 24, 20, 29, 27, 12, 95, 52, 45, 75, 80, 21, 30
 - c. 30, 12, 52, 21, 5, 25, 27, 95, 45, 80, 1, 34, 20, 29, 75, 24
 - d. Tất cả đều sai
20. Cây tìm kiếm nhị phân là cây nhị phân với
- a. Nhân của nút lớn hơn nhân của nút con trái và nhỏ hơn nhân của nút con phải
 - b. Nhân của nút lớn hơn nhân của nút con phải và nhỏ hơn nhân của nút con trái
 - c. Nhân của nút lớn hơn nhân của tất cả các nút ở cây con trái và nhỏ hơn nhân của tất cả các nút ở cây con phải
 - d. Nhân của nút lớn hơn nhân của tất cả các nút ở cây con phải và nhỏ hơn nhân của tất cả các nút ở cây con trái
21. Trong chương trình chính, để thêm phần tử x vào cuối danh sách L ta thực hiện lời gọi như sau:
- a. InsertList (x, Last(L), L);
 - b. InsertList (x, Last(*L), L);
 - c. InsertList (x, EndList(&L), &L);
 - d. InsertList (x, EndList(L), &L); (Lưu ý phân biệt hình thức gọi hàm ở chương trình chính(Hàm main) và ở chương trình con)
22. Khi xóa nút 27 ra khỏi cây tìm kiếm nhị phân, ta thực hiện như sau:



- a. Thay nút 27 bởi nút 24 và quay về xóa nút 24.
 - b. Thay nút 27 bởi nút 20 và quay về xóa nút 20.
 - c. Cho con phải của nút 12 trở xuống nút 20. (Thay 27 thành nút Lớn nhất của cây con Trái nút 27 là 24)
 - d. Cho con phải của nút 12 trở xuống nút 24.
23. Số bước cần duyệt để tìm nút 25 trên cây ở câu 23 theo giải thuật Search(x,T) là:
- a. 3
 - b. 4
 - c. 5
 - d. 6
- (Giải thuật Search kiểm tra từng nút current xem có phải nút cần tìm không, nếu không thì ss xem Lớn hay nhỏ hơn để Định quy duyệt tiếp, trên Graph có 5 nút => 5 lần, sau lần 5 vẫn không phải nút cần tìm nên lúc này current trở đến NULL => Thêm 1 lần đệ quy nữa => Tổng 6 bước duyệt)
24. Xóa nút 52 trên cây ở câu 23, ta cần:
- a. Thay nút 52 bởi nút 37 và quay về xóa nút 37.
 - b. Thay nút 52 bởi nút 40 và quay về xóa nút 40. (Thay 52 thành nút Lớn nhất của cây con Trái nút 52 là 40 - Cũng có thể thay bằng nút Nhỏ nhất của cây con Phải 52 là 75)
 - c. Thay nút 52 bởi nút 80 và quay về xóa nút 80.
 - d. Tất cả đều đúng
25. Thêm nút 42 vào cây ở câu 22, ta thực hiện:
- a. Không thêm được
 - b. Gắn nút 42 vào con phải của 37.

- c. Gắn nút 42 vào con trái của 75.
d. Gắn nút 42 vào con phải của 40.
26. Hãy cho biết thao tác nào không được phép dùng trên cấu trúc ngăn xếp
- Thêm một phần tử vào đỉnh ngăn xếp.
 - Xóa một phần tử ở vị trí bất kì khỏi ngăn xếp. (Chỉ thao tác với phần tử đầu Stack)
 - Thêm một phần tử vào vị trí bất kì trong ngăn xếp. (Như câu B)
 - Cả b và c.**
27. Cho biểu thức trung tố: $(a * (5 + b) - 2 * c) + 4$, biểu thức nào sau đây là dạng hậu tố của nó ? (Biểu thức trung tố giống như biểu thức thường nên khi vẽ chỉ cần:
- Thực hiện ngược lại từ phải sang trái: $(...) + 4 \Rightarrow$ Root là dấu $(+)$
 - Ưu tiên ngoài ngoặc trước rồi mới tới trong ngoặc (trong ngoặc là nhánh con)
- \Rightarrow Tóm lại là ngược lại logic thường là được!)
- $a \ 5 \ * \ b \ + \ 2 \ - \ c \ * \ 4 \ +$
 - $a \ 5 \ b \ + \ * \ 2 \ - \ c \ * \ 4 \ +$
 - $a \ 5 \ b \ + \ * \ 2 \ c \ * \ - \ 4 \ +$**
 - Tất cả đều sai
28. Cho danh sách liên kết
- 
- Ssau khi thực hiện tập lệnh
- ```

p=First(Header);
while (p->next!=q){
 printf("%c ",p->next->data);
 p=p->next;
}

```
- Kết quả trên màn hình là:
- X Y Z T U
  - X Y Z T
  - X Y Z**
  - Y Z T
29. Cho bảng băm đóng với số bucket  $B=10$  và hàm băm  $h(x) = x \bmod B$  và giải quyết đựng độ bằng phương pháp băm lại bình phương ( $h(x) = (x+i^2) \bmod B$ ). Kết quả bảng băm sau khi thực hiện các thao tác thêm 3, 5, 9, 12, 15, 16, 25 là:

Giải:

$3 \Rightarrow [3]$ ,  $5 \Rightarrow [5]$ ,  
 $9 \Rightarrow [9]$ ,  $12 \Rightarrow [2]$ ,  
 $15 \Rightarrow [5]$  (trùng)  $\Rightarrow [5 + 1^2] \Rightarrow [6]$ ,  
 $16 \Rightarrow [6]$  (trùng)  $\Rightarrow [6 + 1^2] = [7]$ ,  
 $25 \Rightarrow [5]$  (trùng)  $\Rightarrow [5 + 1^2] \Rightarrow [6]$  (trùng)  $\Rightarrow [5 + 2^2] \Rightarrow [9]$  (trùng)  
 $\Rightarrow [5 + 3^2] \Rightarrow [4]$   
 Tổng kết:  $[0], [1], [2], [3], [4], [5], [6], [7], [8], [9]$

: [-], [-], [12], [3], [25], [5], [15], [16], [-], [9]

a)            b)            **c)**            d)

30. Danh sách duyệt hậu tự(LRN) của cây nhị phân cho bởi cặp danh sách duyệt sau  
NLR: A, B, C, D, G, E, F (A là Root => LRN thì A đứng cuối, còn B và C)  
và LNR: C, B, D, G, A, E, F ( E là con phải của F =>) là:  
(Tương tự cách giải câu 34: Độ quy chia nhỏ vấn đề ra rồi giải)

a. C, B, G, E, D, A, F

**b. C, G, D, B, F, E, A**

c. C, G, D, B, E, F, A

d. Cặp danh sách đã cho không hợp lệ.

31. Danh sách duyệt một cây nhị phân theo phương pháp duyệt trung tự nhị phân và trung tự tổng quát luôn luôn khác nhau khi và chỉ khi

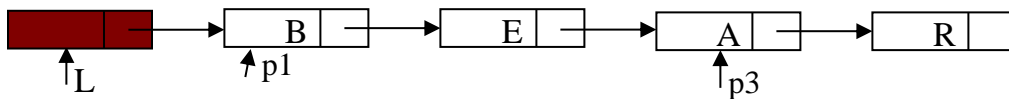
a. Có tồn tại nút bị khuyết con trái trên cây

b. Có tồn tại nút bị khuyết con phải trên cây

**c. Có tồn tại nút có con phải và bị khuyết con trái trên cây.**

d. Có tồn tại nút có con trái và bị khuyết con phải trên cây

32. Cho danh sách liên kết sau :



p3->Next->Next = p1->Next;

L->Next = p3->Next;

p1->Next = NULL;

p3->Next = p1;

p1->Element=Retrieve(p3,L);

PrintList(L); // Hàm in danh sách ra màn hình

Kết quả hiển thị trên màn hình là:

a. REAB

b. BEAR

**c. REAR**

d. Một danh sách khác

33. Khi thêm phần tử vào hàng cài đặt bằng mảng vòng không rỗng thì:

a. Front luôn luôn không đổi và Rear luôn luôn thay đổi

**b. Front luôn luôn không đổi (do emptyQueue không có nên không thay đổi Front) và Rear có thể thay đổi**

c. Front có thể thay đổi và Rear luôn luôn thay đổi

d. Front có thể thay đổi và Rear có thể thay đổi

34. Danh sách duyệt trung tự tổng quát của cây nhị phân cho bởi hai danh sách

LRN: 5, 6, 4, 2, 9, 10, 8, 7, 3, 1 và LNR: 2, 5, 4, 6, 1, 7, 9, 8, 10, 3 là

- a. 5, 4, 2, 6, 1, 9, 8, 10, 3, 7
  - b. 5, 4, 6, 2, 1, 9, 8, 10, 7, 3 (Giải theo kiểu đệ quy, tìm root -> chia nhỏ Left và Right thành 2 cây con -> Tìm Root tiếp -> ... đến khi hết cây con chỉ còn các nút lá: Kinh nghiệm, tìm được root nào thì thêm vào cây, xóa nút đó trong ds duyệt để khỏi rối mắt)
  - c. 2, 5, 4, 6, 3, 1, 8, 7, 9, 10
  - d. 4, 5, 6, 2, 1, 3, 7, 9, 8, 10
35. Cây ở câu 35 có số nút trung gian(Trừ Root và leaf) là:
- a. 4
  - b. 5
  - c. 6
  - d. 7
36. Cây ở câu 35 có nút 7 là nút
- a. Nút lá
  - b. Nút trung gian có 2 con
  - c. Nút trung gian có 1 con phải
  - d. Nút trung gian có 1 con trái
37. Chiều cao của cây ở câu 35 là:
- a. 2
  - b. 3
  - c. 4
  - d. 5
38. Hàm sau được thiết kế trong danh sách liên kết với nội dung là:  
Position **NONAME** (ElementType x, List L)  
{ Position p=First(L);  
  while (p!= Endlist(L))  
    if (Retrieve (P,L)==x)     return 1;  
    else     p=Next(p,L);  
  return 0;  
}
- Hàm này thực hiện được không và làm nhiệm vụ gì?
- a. Hàm trả về tìm vị trí phần tử đầu có nội dung là x.
  - b. Hàm trên bị lỗi ở kiểu dữ liệu trả về cho hàm. (Do Position chỉ có dạng trả về là NULL và Địa chỉ node)
  - c. Hàm kiểm tra xem có phần tử X trong danh sách hay không.
  - d. Hàm luôn trả về giá trị 0.
39. Nguyên tắc làm việc của ngăn xếp là:
- a. FIFO
  - b. FILO
  - c. LILO
  - d. Câu a và c cùng đúng.
40. Cây AVL là cây gì?
- a. Cây cân bằng hoàn toàn
  - b. Cây cân bằng tương đối
  - c. Cây tìm kiếm nhị phân cân bằng hoàn toàn.
  - d. Cây tìm kiếm nhị phân cân bằng tương đối. (Cây AVL là cây BST mà chiều cao của hai cây con của mọi nút chênh lệch tối đa là 1 = Hay còn gọi như câu D)

Lưu ý khi trắc nghiệm Queue:

Tùy bài cho Queue ban đầu thế nào: Thao tác luôn được thực hiện? => “**Luôn luôn**”. Ngược lại không có điều kiện ban đầu thì Queue có thể rỗng hoặc đầy thì Thêm và Xóa chỉ là “**Có thể**” (Ở dưới đang xét với th tổng quát)

Mảng vòng:

- Thêm: Front có thể thay đổi (Queue rỗng  $\Rightarrow$  Front = 0), Rear có thể thay đổi
- Xóa: Front có thể thay đổi, Rear có thể thay đổi (Nếu Queue còn 1 phần tử thì `makeNull(Q)`)

Mảng Tĩnh tiến:

- Thêm: Front có thể thay đổi (Nếu `Rear == MaxLenght` thì thêm dịch cả Queue lên đầu mảng), Rear có thể thay đổi
- Xóa: Front có thể thay đổi, Rear có thể thay đổi (Nếu `Front > Rear` thì `makeNullQueue`)