# [123doc] - trac-nghiem-mon-cau-truc-du-lieu-va-giai-thuat

Anatomy and physiology. (Antioch University New England)

**ĐỀ THI TRẮC NGHIỆM MÔN CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**
**LỚP CDTH8K (Học kỳ I năm học 2007 – 2008)**
Ngày thi: 26/12/2007
Bộ đề thi: 221
Thời gian: 60 phút (Sinh viên không được sử dụng tài liệu)

***Câu 1.*** Thêm phần tử có vùng giá trị là 44 của cây nhị phân tìm kiếm cân bằng sẽ làm cây mất cân bằng

Sau khi thêm, cây được cân bằng lại

**a.**

b.

c. Cả a, b đều sai.

d. Cả a, b đều đúng.

***Câu 2.*** Tìm mô tả đúng nhất cho hàm *TinhTong* sau

```
int TinhTong(int N)
{       int so = 2;       int tong = 0;      int dem = 0;
        while (dem <N)
        {
                if (KiemTra(so) == 1)
                {
                        tong = tong + so;
                        dem ++;
                }
                so = so + 1;
        }
        return tong;
} Trong đó
int KiemTra(int so)
{
        for (int i = 2; i<so; i++)
                if (so%i == 0)
                        return 0;
        return 1;
}
```

a. Hàm tính tổng N số nguyên đầu tiên
b. Hàm tính tổng N số nguyên tố nhỏ hơn N
**c. Cả a, b đều sai**
d. Cả a, b đều đúng

***Câu 3***. Mối quan hệ giữa cấu trúc dữ liệu và giải thuật có thể minh họa bằng đẳng thức:
**a. Cấu trúc dữ liệu + Giải thuật = Chương trình**
b. Cấu trúc dữ liệu + Chương trình = Giải thuật
c. Chương trình + Giải thuật = Cấu trúc dữ liệu
d. Cấu trúc dữ liệu = Chương trình

***Câu 4.*** Các tiêu chuẩn đánh giá cấu trúc dữ liệu. Để đánh giá một cấu trúc dữ liệu chúng ta thường dựa vào một số tiêu chí
a. Cấu trúc dữ liệu phải tiết kiệm tài nguyên (bộ nhớ trong),
b. Cấu trúc dữ liệu phải phản ảnh đúng thực tế của bài toán,
c. Cấu trúc dữ liệu phải dễ dàng trong việc thao tác dữ liệu.
**d. Cả a, b, c đều đúng**

***Câu 5.*** Đoạn mã giả dưới đây mô tả thuật toán gì?
Thuật toán:
B1: k = 1
B2: IF M[k] == X AND k != N
    B2.1: k++
    B2.2: Lặp lại B2
B3: IF k < N
    Thông báo tìm thấy tại vị trí k
B4: ELSE
    Không tìm thấy.
B5: Kết thúc
a. Tìm nhị phân phần tử có giá trị X
b. Tìm phần tử nhỏ nhất của  mảng M bao gồm N phần tử
**c. Tìm tuyến tính phần tử có giá trị X**
d. Cả a, b, c đều sai

***Câu 6.*** Cho hàm tìm kiếm tuyến tính như sau
```
int TimKiem (int M[], int N, int X)
{       int k = 0;
        M[N] = X;
        while (M[k] != X)
                k++;
        if (k < N)
                return (k);
        return (-1);
}
```
Chọn câu đúng nhất:
a. Hàm sẽ trả về 0 nếu không tìm thấy phần tử có giá trị là X
b. Hàm sẽ trả về 1 nếu tìm thấy phần tử có giá trị lả X
**c. Hàm sẽ trả về -1 nếu không tìm thấy phần tử có giá trị là X**
d. Hàm sẽ trả về 1 nếu không tìm thấy phần tử có giá trị lả X

***Câu 7***. Xét thủ tục sau:
```
int TimKiemNP (int M[], int First, int Last, int X)
{
        if (First > Last)
                return (-1);
        int Mid = (First + Last)/2;
        if (X == M[Mid])
                return (Mid);
```

```
        if (X < M[Mid])
                return(TimKiemNP (M, First, Mid – 1, X));
        else
                return(TimKiemNP (M, Mid + 1, Last, X));
}
```

Lựa chọn câu đúng nhất để mô tả thủ tục trên

a. Thủ tục hỗ trợ tìm kiếm phần tử có giá trị là X trên mảng các phần tử từ chỉ số từ First đến chỉ số Last
**b. Thủ tục hỗ trợ tìm kiếm đệ quy phần tử có giá trị là X trên mảng các phần tử từ chỉ số từ First đến chỉ số Last**
c. Thủ tục hỗ trợ tìm kiếm đệ quy phần tử có giá trị là X trên mảng các phần tử từ chỉ số từ Last đến chỉ số First
d. Thủ tục hỗ trợ tìm kiếm không đệ quy phần tử có giá trị là X trên mảng các phần tử từ chỉ số từ Last đến chỉ số First

***Câu 8***. Chọn câu đúng nhất để mô tả thuật toán sắp xếp nổi bọt (Bubble Sort) trên mảng M có N phần tử:

**a. Đi từ cuối mảng về đầu mảng, trong quá trình đi nếu phần tử ở dưới (đứng phía sau) nhỏ hơn phần tử đứng ngay trên (trước) nó thì hai phần tử này sẽ được đổi chỗ cho nhau. Sau mỗi lần đi chúng ta đưa được một phần tử trồi lên đúng chỗ. Sau N–1 lần đi thì tất cả các phần tử trong mảng M sẽ có thứ tự tăng.**
b. Đi từ đầu mảng về cuối mảng, trong quá trình đi nếu phần tử ở dưới (đứng phía sau) nhỏ hơn phần tử đứng ngay trên (trước) nó thì hai phần tử này sẽ được đổi chỗ cho nhau. Sau mỗi lần đi chúng ta đưa được một phần tử trồi lên đúng chỗ. Sau N lần đi thì tất cả các phần tử trong mảng M sẽ có thứ tự tăng.
c. Đi từ cuối mảng về đầu mảng, trong quá trình đi nếu phần tử ở dưới (đứng phía sau) nhỏ hơn phần tử đứng ngay trên (trước) nó thì hai phần tử này sẽ được đổi chỗ cho nhau. Sau mỗi lần đi chúng ta đưa được một phần tử trồi lên đúng chỗ. Sau N lần đi thì tất cả các phần tử trong mảng M sẽ có thứ tự tăng.
d. Cả a, b, c đều sai

***Câu 9***. Hàm mô tả sắp xếp nổi bọt (Bubble Sort) trên mảng M có N phần tử

```
void BubbleSort(int M[], int N)                                  [1]
{                                                                [2]
        int Temp;                                                [3]
        for (int I = 0; I < N-1; I++)                            [4]
                ……..                                             [5]
                        if (M[J] < M[J-1])                       [6]
                        {                                        [7]
                                Temp = M[J];                     [8]
                                M[J]  = M[J-1];                  [9]
                                M[J-1] = Temp;                   [10]
                        }                                        [11]
        return;                                                  [12]
}                                                                [13]
```

Lệnh nào sau đây sẽ được đưa vào dòng lệnh thứ [5] của thủ tục

a. for (int J = N-1; J > I; J++)
b. for (int J = N; J < I; J--)
**c. for (int J = N-1; J > I; J--)**
d. Không có dòng lệnh nào phù hợp, không cần thêm vào thuật toán vẫn chạy đúng

***Câu 10***. Thủ tục mô tả thuật toán sắp xếp chọn trực tiếp (Straight Selection Sort):

```
void SapXepChonTrucTiep(T M[], int N)
{       int K = 0, PosMin;
        int Temp;
        while (K < N-1)
        {
                T Min = M[K];
                PosMin = K;
                for (int Pos = K+1; Pos < N; Pos++)
                        if (Min > M[Pos])
                        {
                                Min = M[Pos];
                                PosMin = Pos
```

```
            }
        .....                                                    [1]
        .....                                                    [2]
        .....                                                    [3]
        K++;
    }
    return;
```

} Chọn câu lệnh thích hợp để đưa vào [1], [2], [3] với mục tiêu hoán vị M[K] và M[PosMin]

a. Temp = M[K] ;
   Temp = M[PosMin];
   M[PosMin] = Temp;

b. M[K] = Temp;
   M[K] = M[PosMin];
   M[PosMin] = Temp ;

c. Temp = M[K] ;
   M[PosMin] = M[K];
   M[PosMin] = Temp ;

**d. Temp = M[K] ;**
**M[K] = M[PosMin];**
**M[PosMin] = Temp ;**

***Câu 11***. Đối với thuật toán sắp xếp chọn trực tiếp cho dãy các phần tử sau (10 pt)

16 60 2 25 15 45 5 30 33 20

Cần thực hiện ..... chọn lựa phần tử nhỏ nhất để sắp xếp mảng M có thứ tự tăng dần.

a. 7 lần
b. 8 lần
**c. 9 lần**
d. 10 lần

***Câu 12***. Thuật toán sắp xếp chèn trực tiếp (Straight Insertion Sort) được mô tả bằng đoạn mã giả như sau:

```
B1: K = 1
B2: IF (K = N)
        Thực hiện BKT
B3: X = M[K+1]
B4: Pos = 1
B5: IF (Pos > K)
        Thực hiện B7
B6: ELSE                        // Tìm vị trí chèn
        B6.1: If (X <= M[Pos])
                Thực hiện B7
        B6.2: Pos++
        B6.3: Lặp lại B6.1
B7: I = K+1
B8: IF (I > Pos)
        B8.1: M[I] = M[I-1]
        B8.2: I--
        B8.3: Lặp lại B8
B9: ELSE
        B9.1: M[Pos] = X
        B9.2: K++
        B9.3: Lặp lại B2
BKT: Kết thúc
```

Trong đó B8 mô tả trường hợp

a. Nếu còn phải dời các phần tử từ Pos->I về phía sau 1 vị trí
b. Nếu còn phải dời các phần tử từ Pos->K+1 về phía sau 1 vị trí

**c. Nếu còn phải dời các phần tử từ Pos->K về phía sau 1 vị trí**

d. Nếu còn phải dời các phần tử từ Pos->I+1 về phía sau 1 vị trí

***Câu 13***. Giả sử cần sắp xếp mảng M có N phần tử sau theo phương pháp sắp xếp chèn trực tiếp

11 16 12 75 51 54 5 73 36 52 98

Cần thực hiện ..... chèn các phần tử vào dãy con đã có thứ tự tăng đứng đầu dãy M để sắp xếp mảng M có thứ tự tăng dần.

a. 9 lần

**b. 10 lần**

c. 8 lần

d. 7 lần

***Câu 14***. Lựa chọn định nghĩa về danh sách đúng nhất

a. Danh sách là tập hợp các phần tử có kiểu dữ liệu xác định và giữa chúng có một mối liên hệ nào đó.

b. Số phần tử của danh sách gọi là chiều dài của danh sách.

c. Một danh sách có chiều dài bằng 0 là một danh sách rỗng.

**d. Cả a, b, c đều đúng**

***Câu 15***. Tìm mô tả đúng cho hàm sau:

```
int SC (int M[], int Len, int CM[])
{       for (int i = 0; i < Len; i++)
                CM[i] = M[i];
        return (Len);
}
```

a. Hàm thực hiện việc sao chép nội dung mảng CM có chiều dài Len về mảng M có cùng chiều dài. Hàm trả về chiều dài của mảng M sau khi sao chép.

b. Hàm thực hiện việc sao chép nội dung mảng M có chiều dài Len -1 về mảng CM có cùng chiều dài. Hàm trả về chiều dài của mảng CM sau khi sao chép.

c. Hàm thực hiện việc sao chép nội dung mảng CM có chiều dài Len -1 về mảng M có cùng chiều dài. Hàm trả về chiều dài của mảng M sau khi sao chép.

**d. Hàm thực hiện việc sao chép nội dung mảng M có chiều dài Len về mảng CM có cùng chiều dài. Hàm trả về chiều dài của mảng CM sau khi sao chép.**

***Câu 16.*** Cấu trúc dữ liệu mảng có các ưu điểm nào

a. Việc thêm, bớt các phần tử trong danh sách đặc có nhiều khó khăn do phải di dời các phần tử khác đi qua chỗ khác.

b. Việc truy xuất và tìm kiếm các phần tử của mảng là dễ dàng vì các phần tử đứng liền nhau nên chúng ta chỉ cần sử dụng chỉ số để định vị vị trí các phần tử trong danh sách (định vị địa chỉ các phần tử);

c. Mật độ sử dụng bộ nhớ của mảng là tối ưu tuyệt đối

**d. Câu a, b, c đúng**

***Câu 17.*** Định nghĩa nào là đúng với danh sách liên kết

a. Danh sách liên kết là cấu trúc dữ liệu dạng cây.

b. Danh sách liên kết là cấu trúc dữ liệu tự định nghĩa.

**c. Danh sách liên kết là tập hợp các phần tử mà giữa chúng có một sự nối kết với nhau thông qua vùng liên kết của chúng.**

d. Danh sách liên kết là tập hợp các phần tử mà đặt kề cận với nhau trong vùng nhớ.

***Câu 18.*** Định nghĩa cấu trúc dữ liệu của danh sách liên kết đơn được mô tả như sau:

```
typedef struct Node
{       int Key;
        Node * NextNode;
} OneNode;
```

Trong đó, khai báo Node * NextNode; dùng để mô tả

a. Con trỏ trỏ tới phần dữ liệu
**b. Vùng liên kết quản lý địa chỉ phần tử kế tiếp**
c. Con trỏ trỏ tới địa chỉ vùng nhớ của phần tử trước đó trong danh sách liên kết đơn.
d. Con trỏ trỏ tới địa chỉ vùng nhớ của phần tử đầu tiên trong danh sách liên kết đơn.

***Câu 19.*** Với cấu trúc dữ liệu của danh sách liên kết đơn lưu trữ thông tin về phòng máy

```
typedef struct PM
{       int maPM;
        int tongsoMay;
} PHONGMAY;
typedef struct Node
{       PHONGMAY Data;
        Node * NextNode;
} OneNode;
typedef OneNode * SLLPointer;
```

Để quản lý danh sách liên kết đơn bằng phần tử đầu và phần tử cuối, cần định nghĩa kiểu dữ liệu:

a. SLLPointer DanhSach;
**b.**
```
typedef struct SSLLIST
{       SLLPointer First;
        SLLPointer Last;
} LIST;
LIST DanhSach;
```
c.
```
typedef struct SSLLIST
{       SLLPointer First;
        SLLPointer Last;
        int total;
} LIST;
LIST DanhSach;
```
d.
```
typedef struct SSLLIST
{       SLLPointer First;
        int total;
} LIST;
LIST DanhSach;
```

***Câu 20.*** Tổ chức cấu trúc dữ liệu cho danh sách liên kết đơn

```
typedef struct Node
{       int Data;
        Node * Link;
} OneNode;
typedef OneNode * SLLPointer;
```

Mã giả thuật toán thêm một phần tử có giá trị thành phần dữ liệu là NewData vào trong danh sách liên kết đơn SLList vào ngay sau nút có địa chỉ InsNode:

```
B1: NewNode = new OneNode
B2: IF (NewNode = NULL)
        Thực hiện BKT
B3: NewNode ->Link = NULL
B4: NewNode ->Data = NewData
B5: IF (InsNode-> Link = NULL)
        B5.1: InsNode-> Link = NewNode
        B5.2: Thực hiện BKT
// Nối các nút kế sau InsNode vào sau NewNode
B6: ……..
// Chuyển mối liên kết giữa InsNode với nút kế của nó về NewNode
```

B7: ……..
BKT: Kết thúc
B6 và B7 dùng để nối nút kế sau InsNode vào sau NewNode và chuyển mối liên kết giữa InsNode với nút kế nó về NewNode. Hãy chọn câu đúng nhất cho B6 và B7

a. B6: InsNode-> Link = NewNode-> Link
   B7: NewNode = InsNode-> Link
b. B6: InsNode-> Link = NewNode-> Link
   B7: InsNode-> Link = NewNode
c. B6: NewNode-> Link = InsNode-> Link
   B7: NewNode = InsNode-> Link
d. **B6: NewNode-> Link = InsNode-> Link**
   **B7: InsNode-> Link = NewNode**

***Câu 21***. Với định nghĩa cấu trúc dữ liệu cho danh sách liên kết đơn

```
typedef struct Node
{
        int Data;
        Node * Link;
} OneNode;
typedef OneNode * SLLPointer;
```

Hàm dưới đây để thêm một phần tử có giá trị thành phần dữ liệu là NewData vào trong danh sách liên kết đơn SLList vào ngay sau nút có địa chỉ InsNode.

```
SLLPointer ThemGiua(SLLPointer &SList, int NewData, SLLPointer &InsNode)
{       SLLPointer NewNode = new OneNode;

        if (NewNode != NULL)
                NewNode ->NextNode = NULL;
                NewNode ->Data = NewData;
        else
                return (NULL);
        if (InsNode->Link == NULL)
        {
                InsNode-> Link = NewNode;
                return (SList);
        }
        ……
        ……
        return (SList);
}
```

Hãy lựa chọn câu đúng nhất

a. InsNode -> Link = NewNode -> Link;
   InsNode-> Link = NewNode;
b. **NewNode-> Link = InsNode-> Link;**
   **InsNode-> Link = NewNode;**
c. InsNode -> Link = NewNode -> Link;
   NewNode = InsNode-> Link;
d. NewNode-> Link = InsNode-> Link;
   NewNode = InsNode-> Link;

***Câu 22***. Cấu trúc dữ liệu nào tương ứng với LIFO

a. Queue
b. Linked List
c. Tree
d. **Stack**

***Câu 23***. Lựa chọn câu đúng nhất về danh sách liên kết đôi (Doubly Linked List)

a. Vùng liên kết của một phần tử trong danh sách liên đôi có 02 mối liên kết với 01 phần tử khác trong danh sách.
b. Vùng liên kết của một phần tử trong danh sách liên đôi có 01 mối liên kết với 02 phần tử khác trong danh sách.
c. **Vùng liên kết của một phần tử trong danh sách liên đôi có 02 mối liên kết với 02 trước và sau nó trong danh sách.**
d. Vùng liên kết của một phần tử trong danh sách liên đôi có 02 mối liên kết với phần tử đầu và cuối của danh sách.

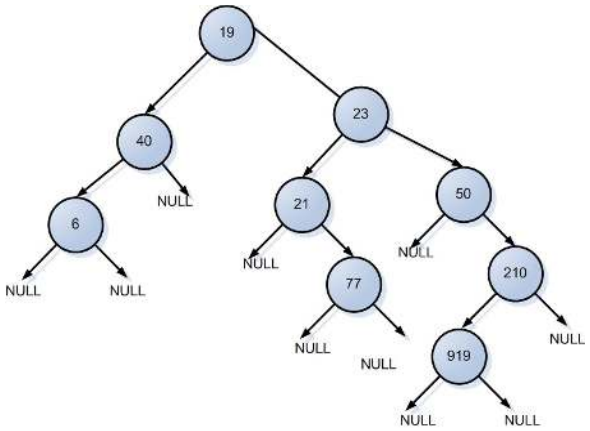
***Câu 24:*** Cho cây nhị phân sau:

Sau khi xoay phải cây, chọn cây đúng nhất

**a.**

**b.**

**c.**




**d.**

***Câu 25:***

Cho thuật toán tìm nhị phân không đệ quy sau

```
int NRecBinarySearch (int M[], int N, int X)
{□      int First = 0;
        int Last = N – 1;
        while (First <= Last)
        {□
```

```
            int Mid = (First + Last)/2;
            if (X == M[Mid])
                  return(Mid);
            if (X < M[Mid])
                  Last = Mid – 1;
            else
                  First = Mid + 1;
      }
      return(-1);
}
```

Chọn câu đúng nhất trong trường hợp tốt nhất khi phần tử ở giữa của mảng có giá trị bằng X:

a. **Số phép gán: Gmin = 3**
   **Số phép so sánh: Smin = 2**
b. Số phép gán: Gmin = 2
   Số phép so sánh: Smin = 3
c. Số phép gán: Gmin = 2
   Số phép so sánh: Smin = 2
d. Số phép gán: Gmin = 0
   Số phép so sánh: Smin = 2

***Câu 26:***

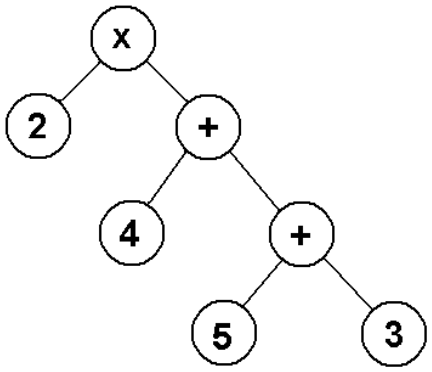
Cho thuật toán sắp xếp Bubble Sort như sau:

```
void BubbleSort(int M[], int N)
{
      for (int I = 0; I < N-1; I++)
            for (int J = N-1; J > I; J--)
                  if (M[J] < M[J-1])
                        Swap(M[J], M[J-1]);
      return;
}
```

Chọn câu đúng nhất cho hàm Swap

a.
```
void Swap(int &X, int &Y)
{     int Temp = X;
      X = Y;
      Y = Temp;
      return;     }
```
b.
```
void Swap(float X, floatY)
{     int Temp = X;
      X = Y;
      Y = Temp;
      return;     }
```
c.
```
void Swap(int *X, int *Y)
{     int Temp = X;
      X = Y;
      Y = Temp;
      return;     }
```
d.
```
void Swap(int X, intY)
{     int Temp = X;
      X = Y;
      Y = Temp;
      return;     }
```

***Câu 27:*** Cho cây biểu thức sau




Chọn biểu thức tương ứng với cây

**a. (2 * (4 + (5 + 3)))**
b. (4 * (2+ (5 + 3)))
c. (2 * (3 + (5 +4)))
d. (2 * (5 + (4+ 3)))

***Câu 28.*** Cho thuật toán sau

```
int LinearSearch (int M[], int N, int X)
{       int k = 0;
        while (M[k] != X   k < N)
                k++;
        if (k < N)
                return (k);
        return (-1);
}
```

Chọn câu đúng nhất trong trường hợp xấu nhất khi không tìm thấy phần tử nào có giá trị bằng X:

a. **Số phép gán: Gmax = 1**
   **Số phép so sánh: Smax = 2N+1**
b. Số phép gán: Gmax = 2
   Số phép so sánh: Smax = 2N+1
c. Số phép gán: Gmax = 1
   Số phép so sánh: Smax = 2N+2
d. Số phép gán: Gmax = 1
   Số phép so sánh: Smax = N+2

***Câu 29.*** Cho thuật toán sau

```
int LinearSearch (float M[], int N, float X)
{
        int k = 0;
        M[N] = X;
        while (M[k] != X) //n+1 lan
                k++;
        if (k < N)
                return (k);
        return (-1);
}
```

Chọn câu đúng nhất trong trường hợp xấu nhất khi không tìm thấy phần tử nào có giá trị bằng X:

a. Số phép gán: Gmax = 1
   Số phép so sánh: Smax = N + 2
**b. Số phép gán: Gmax = 2**
   **Số phép so sánh: Smax = N + 2**
c. Số phép gán: Gmax = 2
   Số phép so sánh: Smax = N + 1
d. Số phép gán: Gmax = 2

Số phép so sánh: Smax =2 N + 2

***Câu 30***. Cho cây nhị phân như sau

Với cách duyệt Inorder (Left – Root – Right) cho ra kết quả là dãy nào? Tìm lựa chọn đúng nhất

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **a.** | **13** | **2** | **55** | **40** | **169** | **11** | **20** | **34** | **50** | **90** | **22** |
| b. | 13 | 2 | 169 | 20 | 11 | 40 | 55 | 22 | 90 | 50 | 34 |
| c. | 34 | 55 | 2 | 13 | 40 | 11 | 169 | 20 | 50 | 90 | 22 |

d. Cả a, b, c đều sai.

***Câu 31.*** Cấu trúc dữ liệu cho kiểu dữ liệu sinh viên như sau

```
typedef struct tagSV{
   char  MSSV[8];
   char  Ten[30];
   char  NgaySinh[11];
   float DTB;
}SV;
```

Khai báo

```
SV sv1, *sv2;
```

Lựa chọn các câu đúng nhất để gán giá trị cho mã sinh viên của sv1 và sv2

a.
```
sv1.MSSV = “Nguyen Van A”;
sv2.MSSV = “Nguyen Van B”;
```

**b.**
```
sv1.MSSV = “Nguyen Van A”;
sv2->MSSV = “Nguyen Van B”;
```

c.
```
sv1->MSSV = “Nguyen Van A”;
sv2->MSSV = “Nguyen Van B”;
```

d.
```
sv1->MSSV = “Nguyen Van A”;
sv2.MSSV = “Nguyen Van B”;
```

***Câu 32.*** Với thủ tục như sau

```
void operation()
{
        int x,a[10],n;
        int x,m,l,h,flag=0;
        cout<<"Enter the element to be searched:";
```

```
        cin>>x;
        l=0; h=n-1;
        while(l<=h)
        {
                m=(l+h)/2;
                if(x==a[m]) {
                        lag=1; break;
                }
                else if(x>a[m])
                        l=m+1;
                else if(x<a[m])
                        h=m-1;
        }
        if(flag==0)
                cout<<"ABSENT";
        else
                cout<<"PRESENT";
```

} Lựa chọn câu đúng nhất để mô tả thủ tục trên

a. Thủ tục tìm nhị phân phần tử được nhập từ bàn phím, nếu tìm thấy sẽ thông báo ABSENT
**b. Thủ tục tìm nhị phân phần tử được nhập từ bàn phím, nếu không tìm thấy sẽ thông báo ABSENT**
c. Thủ tục tìm tuyến tính phần tử được nhập từ bàn phím, nếu tìm thấy sẽ thông báo ABSENT
d. Thủ tục tìm tuyến tính phần tử được nhập từ bàn phím, nếu không tìm thấy sẽ thông báo ABSENT

***Câu 33.*** Biểu diễn và tổ chức ngăn xếp (Stack) bằng danh sách liên kết giả sử bề mặt của ngăn xếp là đầu danh sách liên kết.

```
typedef struct SElement
{       T Key;
        SElement *Next;
} SOneElement;
typedef struct SOneElement *SSTACK;
SSTACK SSP;
```

*Thêm 1 phần tử vào ngăn xếp (dùng cấu trúc dữ liệu mô tả ở trên)*

B1: NewElement = Khởi tạo nút mới (dùng toán tử new)
B2: if (NewElement == NULL)
        Thực hiện BKT
B3: if (SSP == NULL)
        B3.1: SSP = NewElement
        B3.2: Thực hiện BKT
B4: ……
B5: ……
BKT: Kết thúc
Chọn câu lệnh chính xác cho B4 và B5

**a. B4: NewElement ->Next = SSP**
**SSP = NewElement**
b. B4: SSP = NewElement ->Next
B5: SSP = NewElement
c. B4: SSP = NewElement ->Next
B5: NewElement = SSP
d. B4: NewElement ->Next = SSP
B5: NewElement = SSP

***Câu 34.*** Cấu trúc dữ liệu biểu diễn hàng đợi bằng danh sách liên kết

```
typedef struct QElement
{       T Key;
        QElement *Next;
```

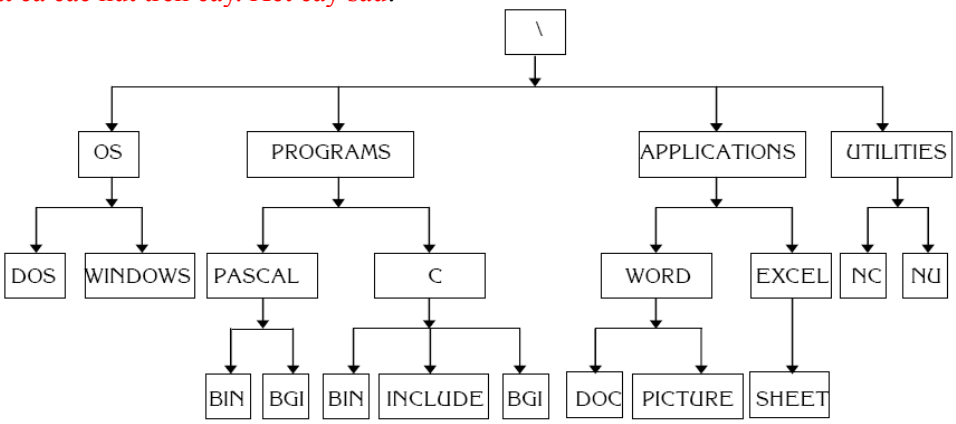
} QOneElement;
typedef QElement  *QType;
Cấu trúc dữ liệu quản lý hàng đợi bằng hai phần tử đầu (Front) và cuối (Rear):
typedef struct QPElement
{	QType Font;
	QType Rear;
} SQUEUE;
SQUEUE  SQList;
Thêm phần tử vào sau phần tử Rear. Giả sử dữ liệu đưa vào hàng đợi là NewData, mã giả được mô tả như sau:
B1: NewElement = Khởi tạo nút mới có thành phần NewData
B2: IF (NewElement == NULL)
	Thực hiện BKT
B3: IF (SQList.Front == NULL) // hàng đợi dang rỗng
	B3.1: SQList.Front = SQList.Rear = NewElement
	B3.2: Thực hiện BKT
B4: ……..
B5: ……..
BKT: Kết thúc
Chọn câu đúng nhất cho bước B4, B5

a. B4: SQList.Front->Next = NewElement
   B5: SQList.Front = NewElement
b. **B4: SQList.Rear->Next = NewElement**
   **B5: SQList.Rear = NewElement**
c. B4: NewElement = SQList.Rear->Next
   B5: SQList.Rear = NewElement
d. B4: NewElement = SQList.Front->Next
   B5: SQList.Font = NewElement

***Câu 35.*** Chọn định nghĩa đúng nhất về hàng đợi (Queue)

a. Hàng đợi còn được gọi là danh sách FILO và cấu trúc dữ liệu này còn được gọi cấu trúc FILO (First In Last Out)
b. **Hàng đợi là một danh sách mà trong đó thao tác thêm 1 phần tử vào trong danh sách được thực hiện 1 đầu này và lấy 1 phần tử trong danh sách lại thực hiện bởi đầu kia.**
c. Hàng đợi là một danh sách mà trong đó thao tác thêm 1 phần tử hay hủy một phần tử trong danh sách được thực hiện 1 đầu.
d. Hàng đợi phải là một danh sách liên kết đơn.

***Câu 36.*** Chiều dài đường đi của một cây (path's length of the tree)  được định nghĩa là tổng tất cả các chiều dài đường đi của tất cả các nút trên cây. Xét cây sau:




a. Chiều dài đường của cây trên là 63
b. Chiều dài đường của cây trên là 64
c. **Chiều dài đường của cây trên là 65**
d. Chiều dài đường của cây trên là 66

***Câu 37.*** Chọn định nghĩa đúng nhất đối với cây nhị phân tìm kiếm

**a. Cây nhị phân tìm kiếm là cây nhị phân có thành phần khóa của mọi nút lớn hơn thành phần khóa của tất cả các nút trong cây con trái của nó và nhỏ hơn thành phần khóa của tất cả các nút trong cây con phải của nó.**

b. Cây nhị phân tìm kiếm là cây nhị phân có thành phần khóa của mọi nút nhỏ hơn thành phần khóa của tất cả các nút trong cây con trái của nó và nhỏ hơn thành phần khóa của tất cả các nút trong cây con phải của nó.

c. Cây nhị phân tìm kiếm là cây nhị phân có thành phần khóa của mọi nút lớn hơn thành phần khóa của tất cả các nút trong cây con trái của nó và lớn hơn thành phần khóa của tất cả các nút trong cây con phải của nó.

d. Cây nhị phân tìm kiếm chính là cây nhị phân

***Câu 38.*** Chọn định nghĩa đúng nhất về cây cân bằng tương đối

a. Cây cân bằng tương đối là một cây nhị phân thỏa mãn điều kiện là đối với mọi nút của cây thì số nút của cây con trái và số nút của cây con phải của nút đó hơn kém nhau không quá 1. Cây cân bằng tương đối còn được gọi là cây AVL (AVL tree).

b. Cây cân bằng tương đối là một cây N phân thỏa mãn điều kiện là đối với mọi nút của cây thì chiều cao của cây con trái và chiều cao của cây con phải của nút đó hơn kém nhau không quá 2. Cây cân bằng tương đối còn được gọi là cây AVL (AVL tree).

**c. Cây cân bằng tương đối là một cây nhị phân thỏa mãn điều kiện là đối với mọi nút của cây thì chiều cao của cây con trái và chiều cao của cây con phải của nút đó hơn kém nhau không quá 1. Cây cân bằng tương đối còn được gọi là cây AVL (AVL tree).**

d. Cây cân bằng tương đối cũng là cây cân bằng hoàn toàn.

***Câu 39***: Định nghĩa cấu trúc dữ liệu của danh sách liên kết đơn được mô tả như sau:

```
struct Node
{
        int Key;
        Node * NextNode;
} OneNode;
```

Trong đó, khai báo Node * NextNode; dùng để mô tả

a. Con trỏ trỏ tới phần dữ liệu

**b. Vùng liên kết quản lý địa chỉ phần tử kế tiếp**

c. Con trỏ trỏ tới phần dữ liệu cuối của danh sách

d. Vùng liên kết quản lý địa chỉ phần tử kế tiếp của phần tử cuối

***Câu 40***. Khi cần thêm một phần tử có giá trị thành phần dữ liệu là NewData (là một số nguyên) vào đầu của danh sách liên kết đơn dùng thuật toán có mã giả mô tả như dưới đây.

```
typedef struct Node
{
        int Data;
        Node * NextNode;
} OneNode;
typedef OneNode * SLLPointer;
SLLPointer SSList;
B1: NewNode = new OneNode
B2: IF (NewNode = NULL)
        Thực hiện BKT
B3: NewNode ->NextNode = NULL
B4: NewNode ->Data = NewData
B5: NewNode->NextNode = SLList
B6: SLList = NewNode
BKT: Kết thúc
```

Tìm mô tả chính xác cho B5

a. Chuyển vai trò đứng đầu của NewNode cho SLList

b. Nối NewNode vào sau SLList

c. Chuyển vai trò đứng đầu của SLList cho NewNode

**d. Nối SLList vào sau NewNode**

***Câu 41:*** Tìm kiếm xem trong danh sách liên kết đơn có tồn tại nút có thành phần dữ liệu là SearchData hay không. Thao tác này chúng ta vận dụng thuật toán tìm tuyến tính để tìm kiếm.

```
typedef struct Node
{
        int Data;
        Node * Link;
} OneNode;
typedef OneNode * Pointer;
Pointer SSList;  // Quản lý danh sách liên kết đơn bởi 1 phần tử đầu
B1: CurNode = SLList
B2: IF (……)
        Thực hiện BKT
B3: CurNode = CurNode->Link
B4: Lặp lại B2
BKT: Kết thúc
```

Chọn điều kiện hợp lý cho mã giả ở B2

a. CurNode != NULL OR CurNode->Data = SearchData
b. CurNode = NULL OR CurNode->Data != SearchData
**c. CurNode = NULL OR CurNode->Data = SearchData**
d. CurNode != NULL OR CurNode->Data != SearchData

***Câu 42:*** Cho cấu trúc dữ liệu như sau:

```
typedef struct Node
{
        int Key;
        Node *NextNode;
} OneNode;
typedef SLLOneNode * Type;
```

Thuật toán chọn trực tiếp viết trên ngôn ngữ C++ áp dụng cho danh sách liên kết đơn quản lý bởi một phần tử đầu tiên được mô tả:

```
void StraightSelection(Type &SList)
{
                Type MinNode;
                int Temp;
                Type CurrNode,TempNode;
                CurrNode = SList;
                while (CurrNode!=NULL)
                {
                        TempNode = CurrNode->NextNode;
                        MinNode = CurrNode;
                        while (TempNode!=NULL)
                        {
                                if (……)
                                        MinNode = TempNode;
                                TempNode = TempNode->NextNode;
                        }
[1]                     Temp = MinNode->Key;
[2]                     MinNode->Key = CurrNode->Key;
[3]                     CurrNode->Key = Temp
                        CurrNode=CurrNode->NextNode;
                }
}
```

Tìm mô tả chính xác cho [1], [2], [3]

a. Hoán vị 2 mối liên kết

**b. Hoán vị 2 vùng giá trị**

c. Hoán vị nút đầu và nút cuối

d. Hoán vị 2 nút kế tiếp nhau

***Câu 43.*** Với cấu trúc dữ liệu như sau

```
typedef struct DNode
{
        int Key;
        DNode * NextNode;
        DNode * PreNode;
} DOneNode;
typedef DLLOneNode * DPointerType;
typedef struct DPairNode
{       DPointerType    DLLFirst;
        DPointerType    DLLLast;
} DPType;
```

Hàm thêm phần tử vào cuối danh sách liên kết đôi quản lý bởi 2 phần tử đầu và cuối

```
DPointerType DLLAddLast(DPType &DList, int NewData)
{
        DPointerType NewNode = gọi hàm tạo nút mới có vùng dữ liệu là NewData;
        if (NewNode == NULL)
                return (NULL);
        if (DList.DLLLast == NULL)
                DList.DLLFirst = DList.DLLLast  = NewNode;
        else
        {
                …….
                …….
                …….
        }
        return (NewNode);
} Hãy lựa chọn câu đúng nhất để điền vào chỗ trống ở trên
```

a.
```
DList.DLLLast ->NextNode = NewNode;
NewNode ->PreNode = DList.DLLLast;
NewNode = DList.DLLLast;
```

b.
```
DList.DLLLast ->NextNode = NewNode;
DList.DLLLast = NewNode ->PreNode;
DList.DLLLast = NewNode;
```

c.
```
NewNode = DList.DLLLast ->NextNode;
NewNode ->PreNode = DList.DLLLast;
DList.DLLLast = NewNode;
```

**d.**
```
DList.DLLLast ->NextNode = NewNode;
NewNode ->PreNode = DList.DLLLast;
DList.DLLLast = NewNode;
```

***Câu 44:*** Với cấu trúc dữ liệu như sau

```
typedef struct DNode
{
        int Key;
        DNode * NextNode;
        DNode * PreNode;
} DOneNode;
typedef DOneNode * DPointerType;
typedef struct DLLPairNode
{       DPointerType    DLLFirst;
```

```
        DPointerType     DLLLast;
} DLLPType;
```
Hàm duyệt qua các nút trong danh sách liên kết đôi quản lý bởi hai địa chỉ nút đầu tiên và nút cuối cùng thông qua DList để xem nội dung thành phần dữ liệu của mỗi nút.
```
void DLLTravelling (DLLPType DList)
{
        DPointerType CurrNode = DList.DLLFirst;
        while (CurrNode != NULL)
        {
                cout << CurrNode->Key;
                ……
        }
        return;
```
} Chọn câu chính xác điền vào chỗ trống để mô tả việc di chuyển từ nút này sang nút khác

a. **CurrNode = CurrNode ->NextNode ;**
b. CurrNode = CurrNode ->Key ;
c. CurrNode ->NextNode = CurrNode;
d. CurrNode ->Key = CurrNode;

***Câu 45:*** Với cấu trúc dữ liệu mô tả cho Stack
```
typedef struct SElement
{
        int Key;
        SElement *Next;
} SOneElement;
typedef SOneElement *SSTACK;
```
Tìm mô tả chính xác cho hàm sau:
```
void SSDelete (SSTACK &SList)
{
        while (SList != NULL)
        {
                SSTACK TempElement = SList;
                SList = SList ->Next;
                TempElement ->Next = NULL;
                delete TempElement;
        }
}
```
a. Hủy phần tử đầu của Stack
b. Hủy phần tử cuối của Stack
c. Hủy phần tử cuối của Stack và lấy giá trị đó in ra màn hình
**d. Hủy toàn bộ Stack**