ĐỀ CƯƠNG ÔN TẬP MÔN CẤU TRÚC DỮ LIỆU & GIẢI THUẬT

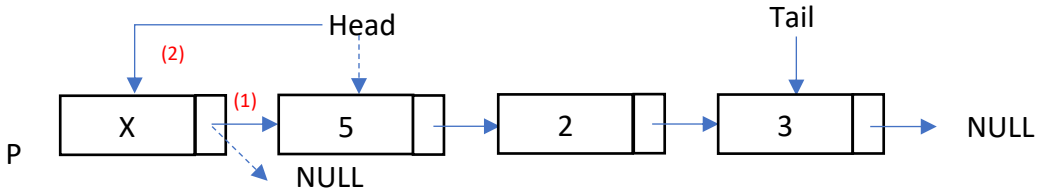
Biên soạn: Nguyễn Phúc Bảo Nhân

Phần 1: Danh sách liên kết đơn

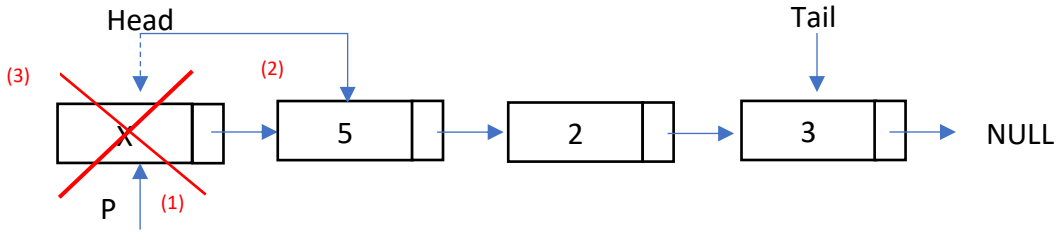
Câu 1:

Vẽ hình minh hoạ các trường hợp sau:

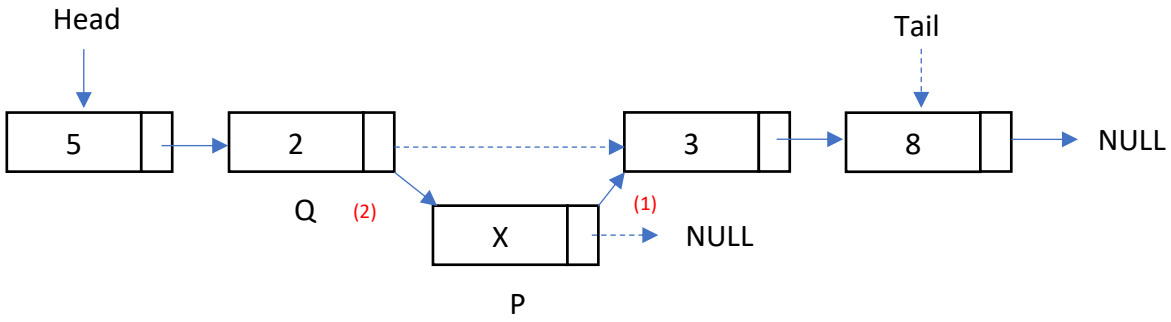
a. Thêm node đầu vào danh sách liên kết



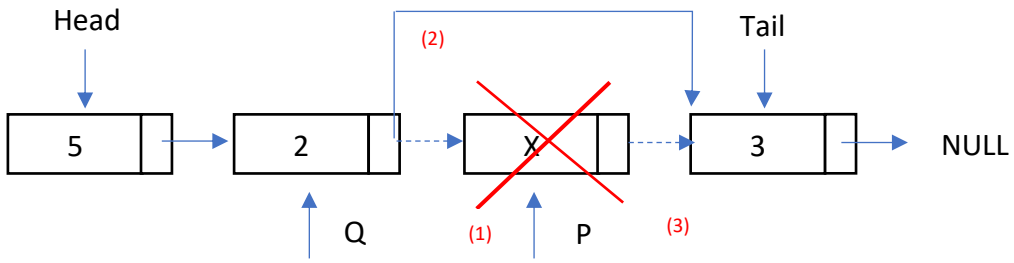
b. Xoá node đầu trong danh sách liên kết



c. Thêm một node p đằng sau node q trong danh sách liên kết đơn



d. Xoá một node p sau một node q trong danh sách liên kết đơn



Câu 2: Cho danh sách liên kết đơn lưu các số nguyên. Viết các hàm:

a. Xuất các số dương là bội của 5 trong danh sách.

```
void xuatSoDuongLaBoiCua5( Slist &lst )
{
	if( lst.head == NULL)
	{
		cout << "\nDanh sách của bạn không có phần tử!" << endl;
		return;
	}
	cout << "\nCác số dương là bội của 5 trong danh sách là: ";
	for(Node *p = lst.head; p != NULL; p = p->next)
	{
		if( p->info % 5 == 0)
			cout << p->info << "\t";
	}
}
```

b. Đếm các số nguyên tố trong danh sách liên kết

```
bool kiemTraSNT(int x)
{
	if(x < 2)
		return false;
	for(int i = 2; i <= sqrt(x); i++)
		if(x % i == 0)
			return false;
	return true;
}

int demCacSNT( Slist &lst )
{
	int dem = 0;
	for( Node* p = lst.head; p != NULL; p = p->next)
		if( kiemTraSNT(p->info) )
			dem++;
	return dem;
}
```

c. Tổng các số chính phương trong danh sách liên kết

```
bool kiemTraSoChinhPhuong( int x )
{
	int temp = sqrt(x);
	if( pow(temp, 2) == x)
		return true;
	return false;
}
```

```
int tinhTongCacSoChinhPhuong( Slist &lst )
{
	int tong = 0;
	for( Node* p = lst.head; p != NULL; p = p->next)
		if( kiemTraSoChinhPhuong(p->info) )
			tong += p->info;
	return tong;
}
```

d. Tìm node có giá trị x (vị trí đầu tiên). Tìm thấy trả về con trỏ đến node có giá trị bằng x, ngược lại trả về NULL.

```
Node *searchSList( Slist &lst, int x)
{
	for( Node* p = lst.head; p != NULL; p = p->next)
		if( p->info == x )
			return p;
	return NULL;
}
```

e. Tìm node có giá trị x (vị trí cuối cùng). Tìm thấy trả về con trỏ đến node có giá trị bằng x, ngược lại trả về NULL.

```
int demSoLuongPhanTuX( int x )
{
	int dem = 0;
	for( Node* p = lst.head; p != NULL; p = p->next)
		if( p->info == x )
			dem++;
	return dem;
}

Node *searchSList( Slist &lst, int x)
{
	int dem = 0;
	for( Node* p = lst.head; p != NULL; p = p->next)
		if( p->info == x )
		{
			dem++;
			if( demSoLuongPhanTu(x) == dem)
				return p;
		}
	return NULL;
}
```

f. Tìm node có giá trị là số hoàn hảo trong danh sách

```
bool kiemTraSoHoanHao( int x )
{
	int tong = 0;
	for(int i = 1; i < x; i++)
		if(x % i == 0)
			tong += i;
	if( tong == x)
		return true;
	return false;
}

Node *timNodeCoGiaTriLaSoHoanHao( Slist &lst )
{
	for( Node* p = lst.head; p != NULL; p = p->next)
		if( kiemTraSoHoanHao(p->info) )
			return p;
	return NULL;
}
```

g. Tìm node có giá trị là số nguyên âm lớn nhất trong danh sách.

```
int timSoNguyenAmLonNhat( Slist &lst )
{
	int max = 0;
	Node* p;
	for( p = lst.head; p != NULL; p = p->next)
		if( p->info < 0 )
		{
			max = p->info;
			break;
		}
	for( p = lst.head; p != NULL; p = p->next)
		if( p->info < 0 && p->info > max)
			max = p->info;
	return max;
}

Node *timNodeCoGiaTriNguyenAmLonNhat( Slist &lst )
{
	int max_nguyen_am = timSoNguyenAmLonNhat(lst);
	for( Node* p = lst.head; p != NULL; p = p->next)
		if( p->info == max_nguyen_am )
			return p;
	return NULL;
```

```
}
```

h. Tìm node có giá trị là số nguyên dương nhỏ nhất trong danh sách.

```
int timSoNguyenDuongNhoNhat( Slist &lst )
{
	int min = 0;
	Node* p;
	for( p = lst.head; p != NULL; p = p->next)
		if( p->info > 0 )
		{
			min = p->info;
			break;
		}
	for( p = lst.head; p != NULL; p = p->next)
		if( p->info > 0 && p->info < min)
			min = p->info;
	return min;
}

Node *timNodeCoGiaTriNguyenDuongNhoNhat( Slist &lst )
{
	int min_nguyen_duong = timSoNguyenDuongNhoNhat(lst);
	for( Node* p = lst.head; p != NULL; p = p->next)
		if( p->info == min_nguyen_duong )
			return p;
	return NULL;
}
```

Phần 3: Cây nhị phân tìm kiếm chứa các số nguyên

Câu 3: Hãy viết các hàm thực hiện các yêu cầu sau:

a. Xuất các node có giá trị là số chẵn, theo thứ tự giảm dần

```
void showTNode(TNode *p)
{
	cout << p->info << "\t";
}
void xuatCacNodeCoGiaTriGiamDan( TNode *root )
{
	if(root == NULL)
		return;
	xuatCacNodeCoGiaTriGiamDan(root->right);
	if(bt.root->info % 2 == 0)
		showTNode(root);
	xuatCacNodeCoGiaTriGiamDan(root->left);
}
```

b. Xuất các node có giá trị là số lẻ, ở mức k

```
void xuatCacNodeCoGiaTriLeOMucK( TNode *root , int k)
{
	if(root == NULL)
		return;
	if( k == 0 && root->info % 2 == 0)
		showTNode(root);
	k--;
	xuatCacNodeCoGiaTriLeOMucK(root->left, k);
	xuatCacNodeCoGiaTriLeOMucK(root->right, k);
}
```

c. Xuất các node chẵn, có con, ở mức k, theo thứ tự giảm dần

```
void xuatCacNodeChanCoConOMucKGiamDan( Tnode *root, int k)
{
	if(root == NULL)
		return;
	if( k == 0 && (root->left != NULL || root->right != NULL))
		showTNode(root);
	k--;
	xuatCacNodeChanCoConOMucKGiamDan(root->right, k);
	xuatCacNodeChanCoConOMucKGiamDan(root->left, k);
}
```

d. Xuất các node có giá trị là số chẵn, có 1 con, ở các mức lớn hơn k.

```
void xuatCacNodeChanCo1ConOMucLonHonK( TNode *root, int k)
{
	if(root == NULL)
		return;
	if(( k < 0 && (root->left != NULL && root->right == NULL) || (root-
>left == NULL && root->right != NULL) && root->info % 2 == 0)
			showTNode(root);
	k--;
	xuatCacNodeChanCo1ConOMucLonHonK(root->left, k);
	xuatCacNodeChanCo1ConOMucLonHonK(root->right, k);

}
```

e. Đếm số nút chứa giá trị là số chẵn, là nút lá.

```
int demSoNutLaChan( TNode *root )
{
```

```
	if( root == NULL )
		return 0;
	if( root->info % 2 == 0 && root->left == NULL && root->right == NULL)
		return 1 + demSoNutLaChan(root->left) + demSoNutLaChan(root->right);
	return demSoNutLaChan(root->left) + demSoNutLaChan(root->right);
}
```

f. Đếm số nút chứa chứa giá trị là số chẵn, có một con, tại mức k.

```
int demSoNutChanCoMotConOMucK(TNode *root, int k)
{
	if(root == NULL)
		return 0;
	if(root->info % 2 == 0 && (root->left != NULL && root->right == NULL) || (root->left == NULL && root->right != NULL) && k == 0)
	{
		k--;
		return 1 + demSoNutChanCoMotConOMucK(root->left, k) + demSoNutChanCoMotConOMucK(root->right, k);
	}
	return demSoNutChanCoMotConOMucK(root->left, k) + demSoNutChanCoMotConOMucK(root->right, k);
}
```

g. Tính tổng các nút lẻ, có ít nhất một con trong cây.

```
int tinhTongCacNutLeCoItNhat1Con(TNode *root)
{
	if(root == NULL)
		return 0;
	if( root->info % 2 != 0 && (root->left != NULL || root->right != NULL))
		return root->info + tinhTongCacNutLeCoItNhat1Con(root->left) + tinhTongCacNutLeCoItNhat1Con(root->right);
	return tinhTongCacNutLeCoItNhat1Con(root->left) + tinhTongCacNutLeCoItNhat1Con(root->right);
}
```

h. Tính tổng các nút lẻ có 2 con, tại các mức lớn hơn k.

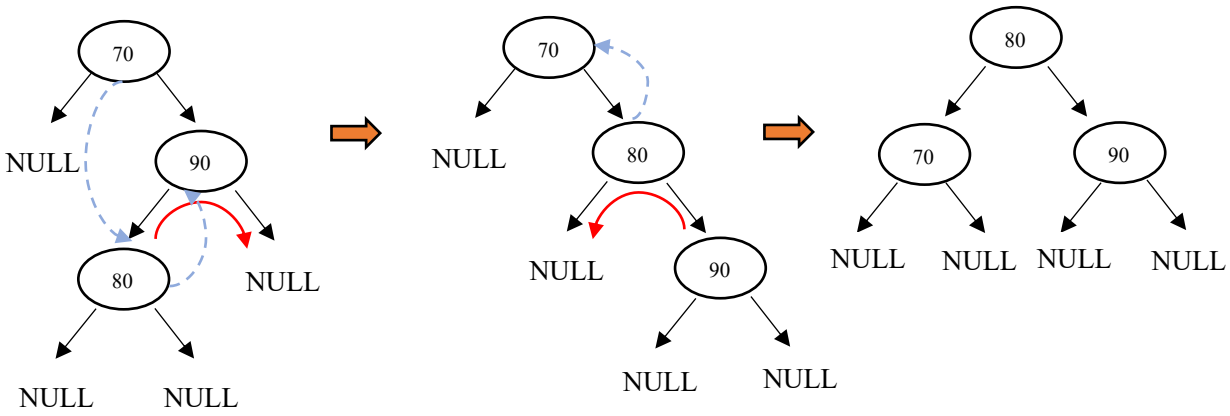
```
int tinhTongNutLeCo2ConTaiMucLonHonK(TNode *root, int k)
{
	if(root == NULL)
		return 0;
```

```
        if( k < 0 && (root->left != NULL && root->right != NULL))
                return root->info +
tinhTongNutLeCo2ConTaiMucLonHonK(root->left, k) +
tinhTongNutLeCo2ConTaiMucLonHonK(root->right, k);
        return tinhTongNutLeCo2ConTaiMucLonHonK(root->left, k) +
tinhTongNutLeCo2ConTaiMucLonHonK(root->right, k);
}
```

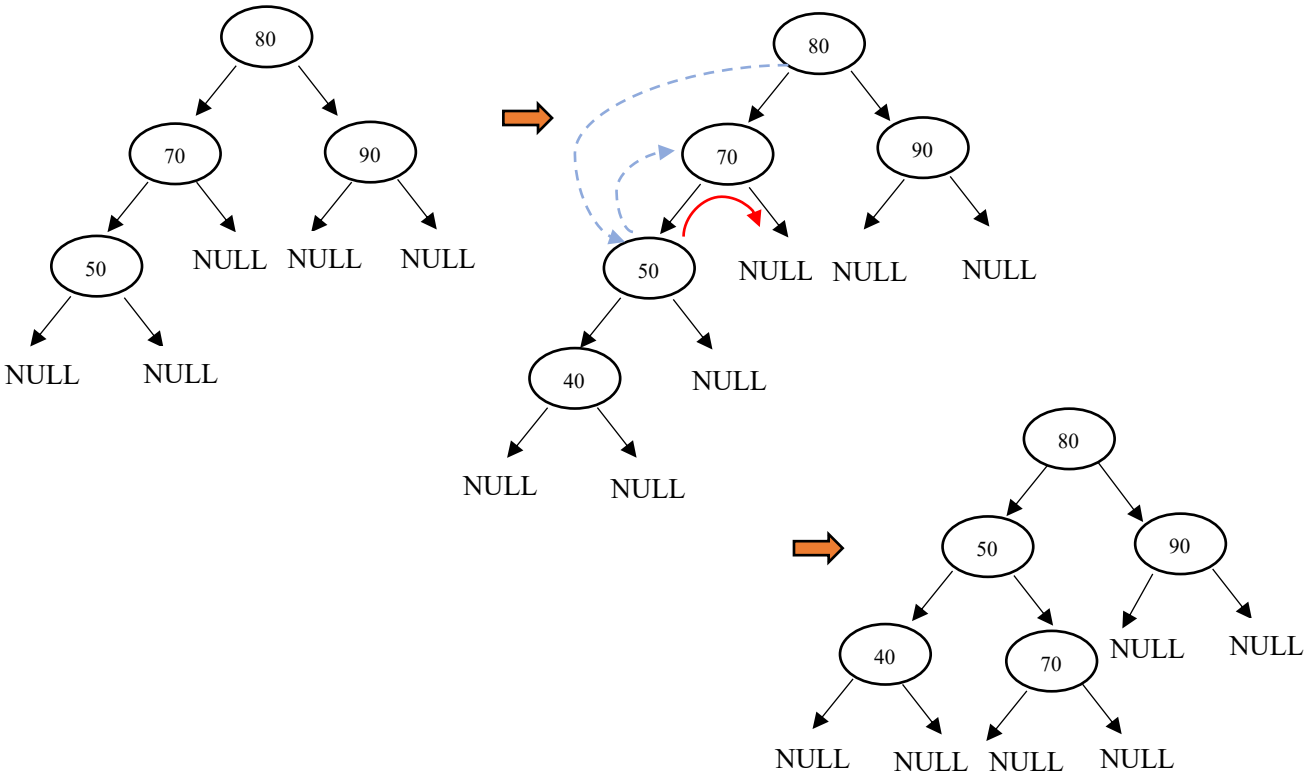
Câu 4: Cho cây T là cây nhị phân tìm kiếm cân bằng (AVL) chứa các số nguyên. Và dãy số A: 70, 90, 80, 50, 40, 75, 60, 65, 68, 67.

a. Tạo cây từ dãy A. Minh hoạ vẽ từng bước quá trình thêm.

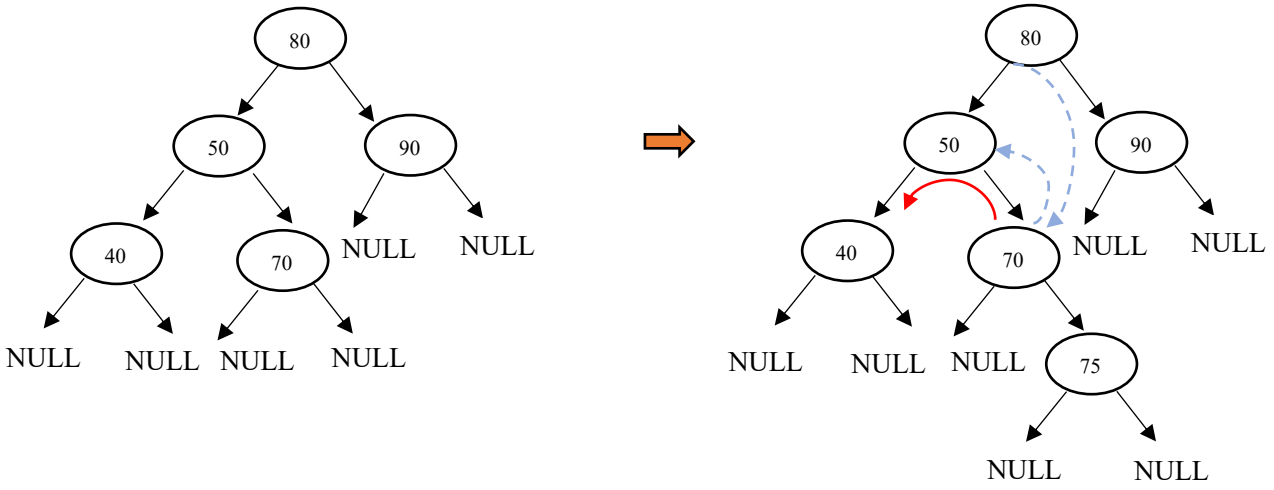
- Thêm các node 70, 90, 80.

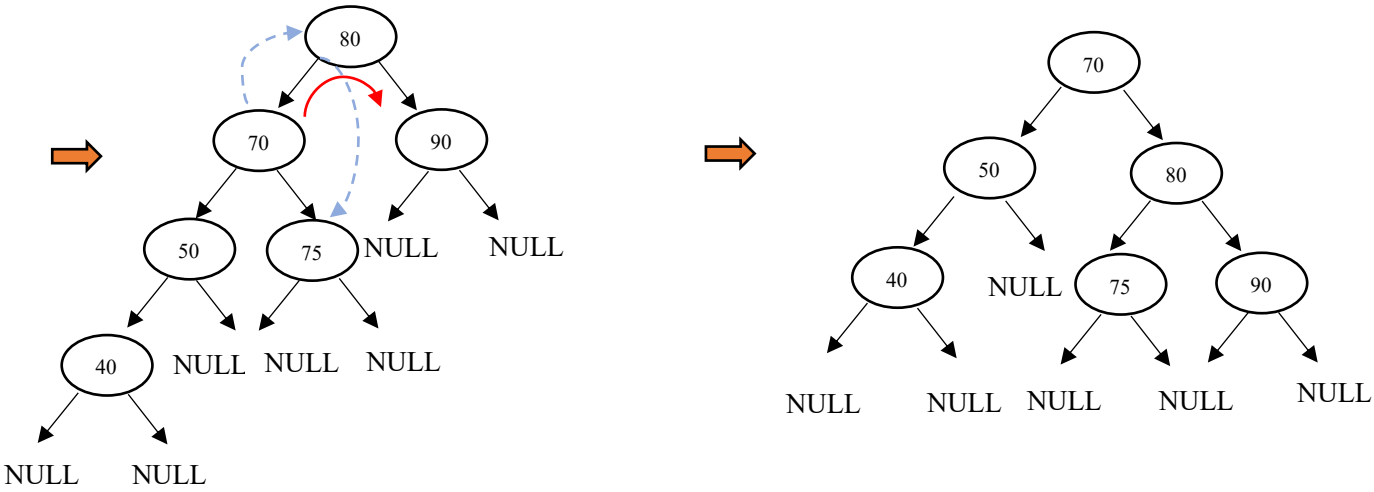


- Thêm các node 50, 40.

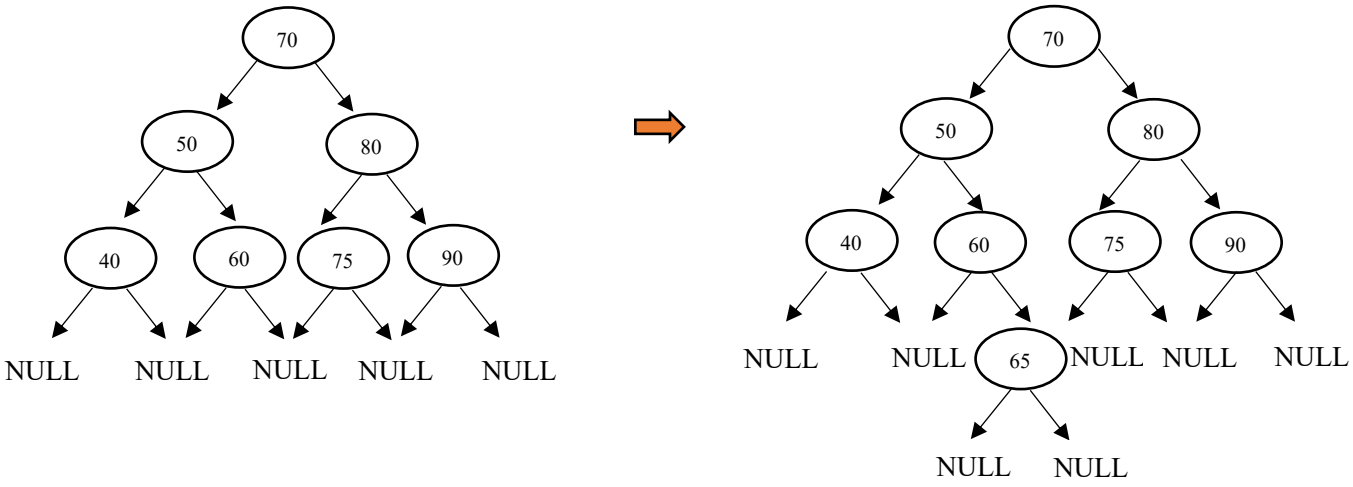


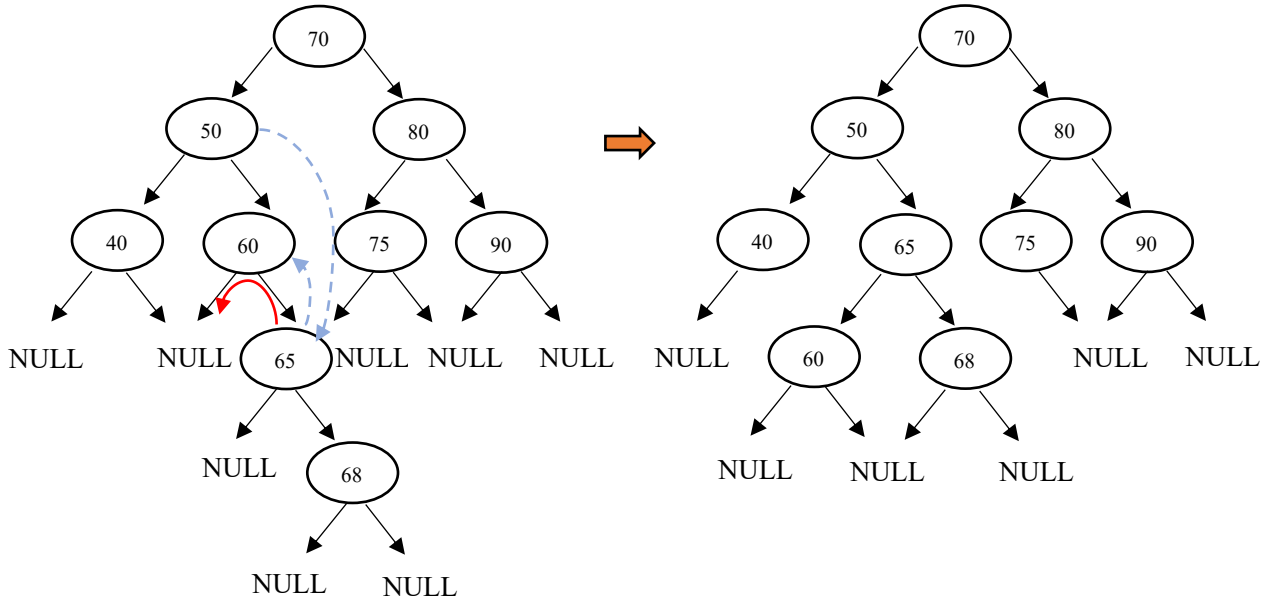
- Thêm node 75.



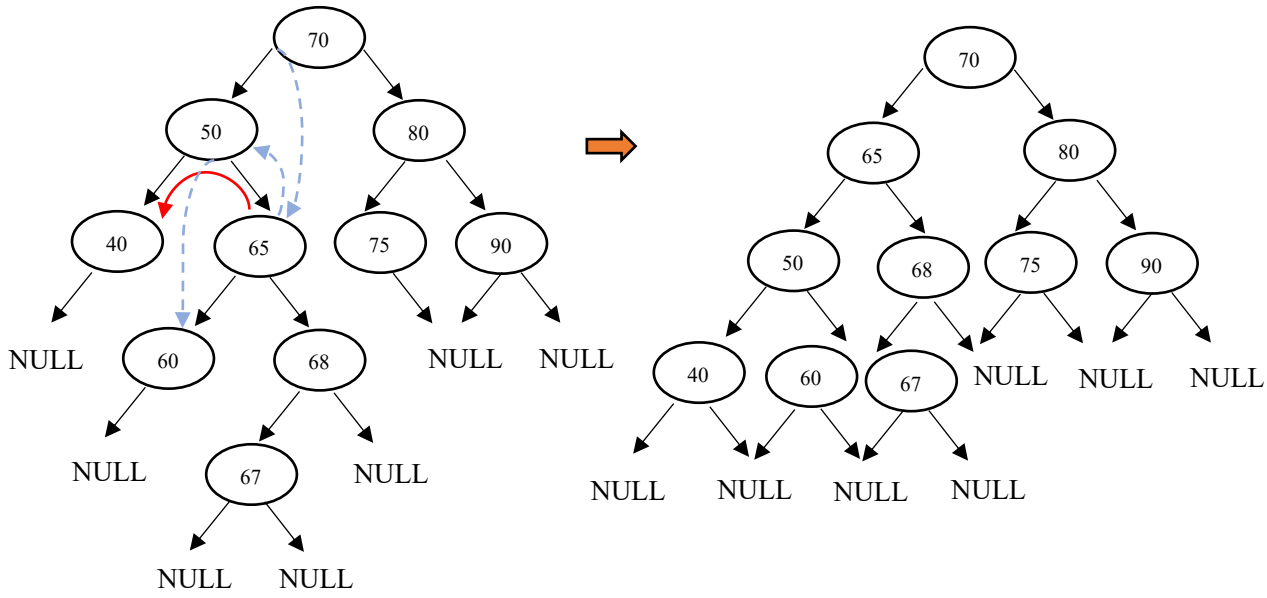


- Thêm các node 60, 65, 68.





- Thêm node 67.

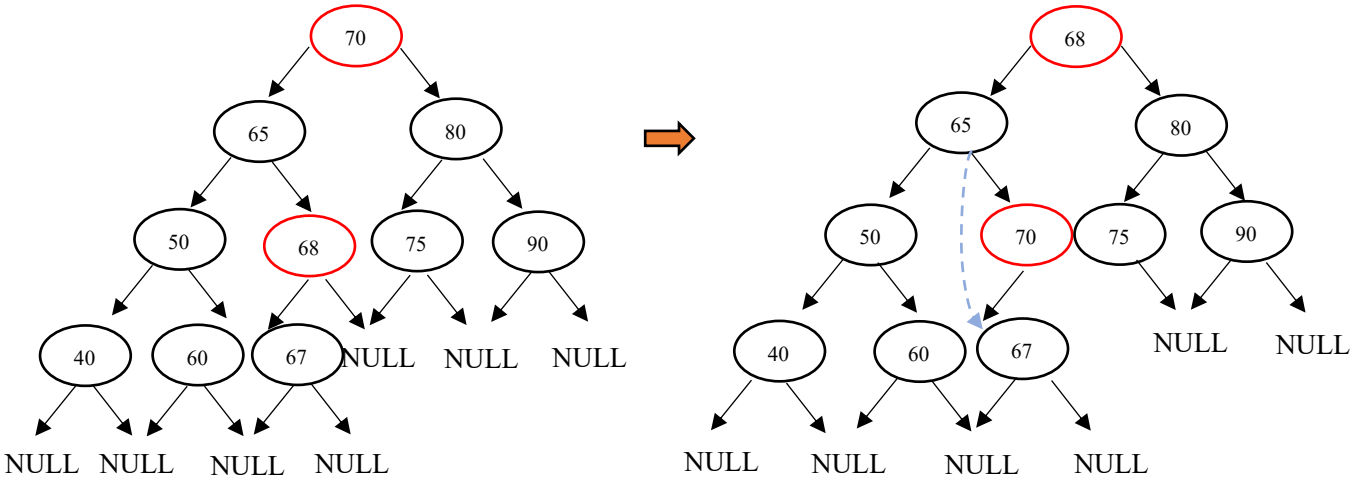


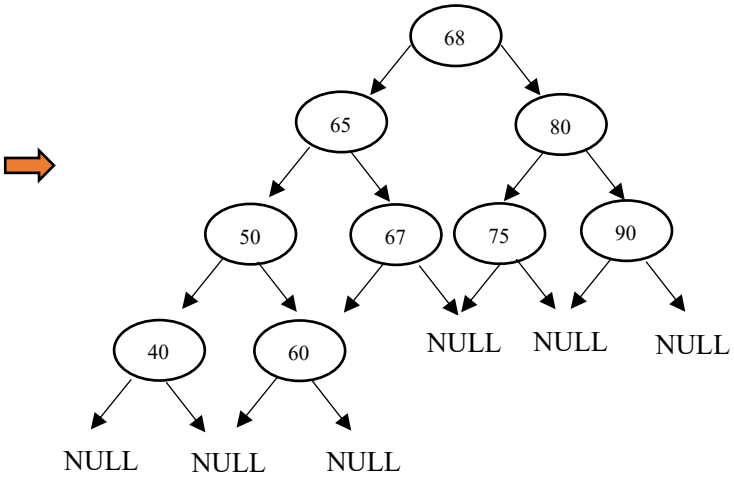
b. Cho biết số lần cần so sánh để tìm phần tử $x_1 = 67$ và $x_2 = 78$.

- Để tìm phần tử $x_1 = 67$ cần 4 lần so sánh.

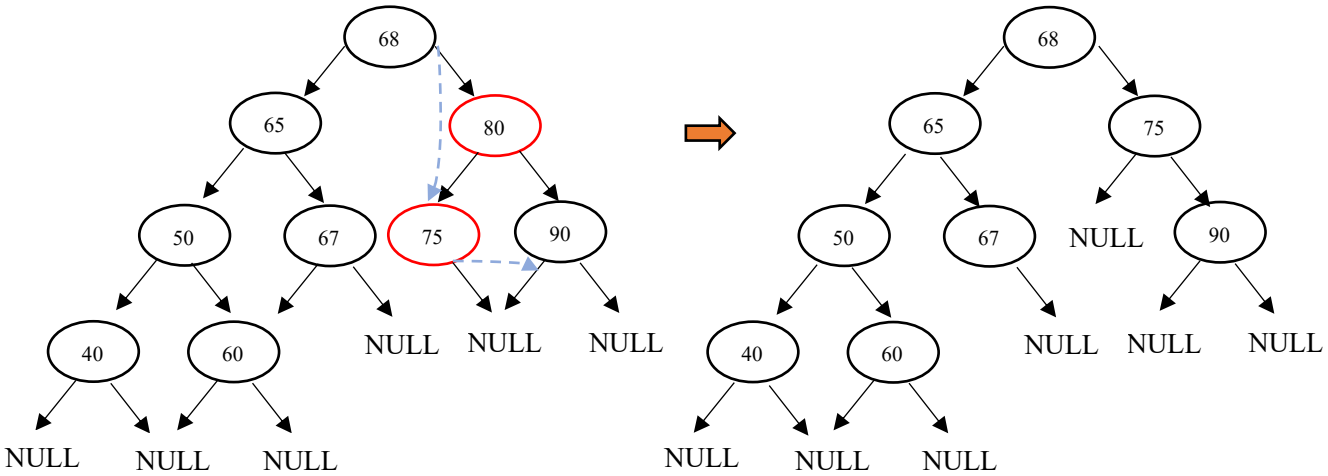
c. Xoá các nút 70, 80, 50, 60, 40, 90. Minh hoạ (vẽ) từng bước quá trình xoá.

- Xoá node 70.

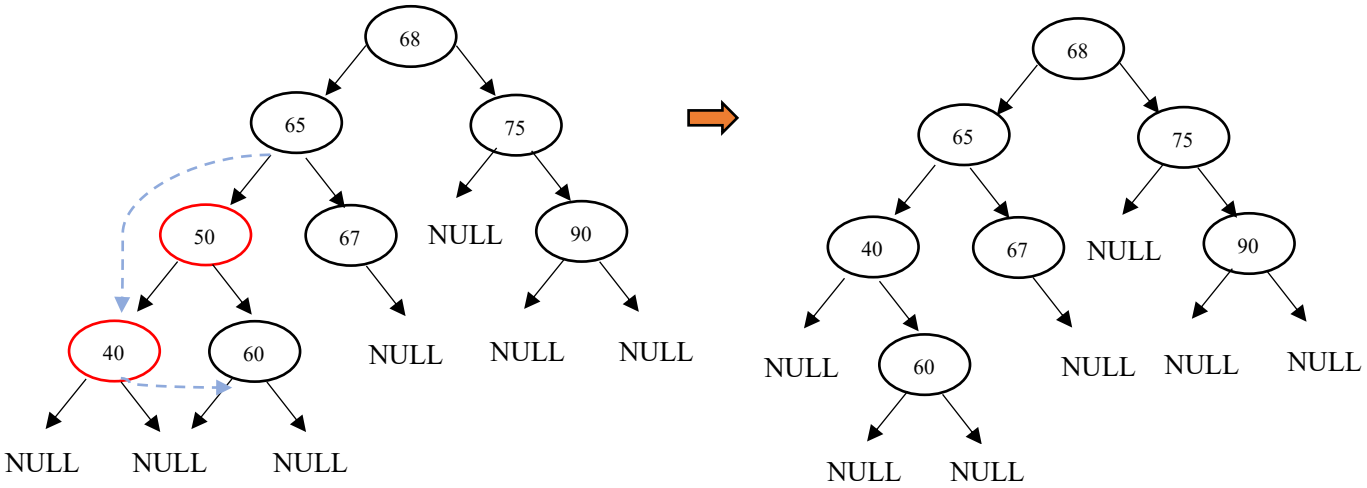




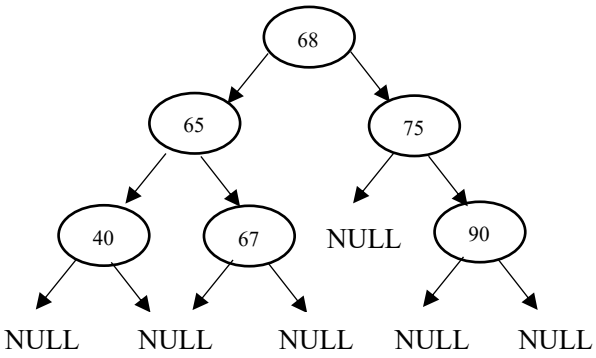
- Xoá node 80.



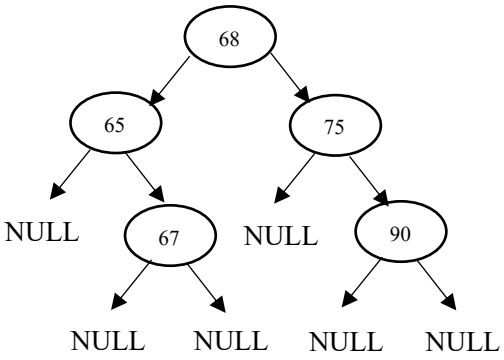
- Xoá node 50.



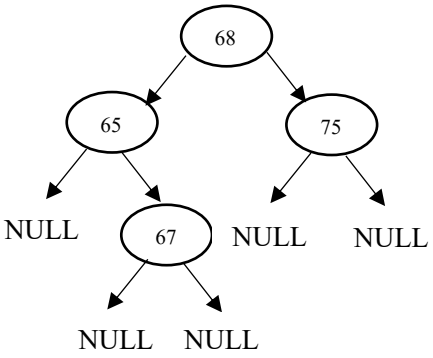
- Xoá node 60.



- Xoá node 40.



- Xoá node 90.



Câu 6: Hãy làm bài tập sau đây:

a. Áp dụng giải thuật chuyển đổi biểu thức dạng trung tố sau (chứa các phép toán +, -, *, /, các toán hạng, dấu ngoặc) sang biểu thức hậu tố bằng cách sử dụng Stack.

P = ((8 + 3 * 2) / 2 + (9 – 3 * 2 + 5) * 2) / 3

Hãy mô tả từng bước theo dạng bảng sau:

STT	Ký tự đang đọc	TT. Stack	BT. Hậu tố
1	(	(	Empty
2	(	((	Empty
3	8	((	8
4	+	((+	8
5	3	((+	83
6	*	((+*	83
7	2	((+*	832
8	)	(	832*+
9	/	(/	832*+
10	2	(/	832*+2
11	+	(	832*+2/
12		(+	832*+2/
13	(	(+(	832*+2/
14	9	(+(	832*+2/9
15	-	(+(-	832*+2/9
16	3	(+(-	832*+2/93
17	*	(+(-*	832*+2/93
18	2	(+(-*	832*+2/932
19	+	(+(-	832*+2/932*
20		(+(	832*+2/932*-
21		(+(+	832*+2/932*-
22	5	(+(+	832*+2/932*-5

23	)	(+	832*+2/932*-5+
24	*	(+*	832*+2/932*-5+
25	2	(+*	832*+2/932*-5+2
26	)	Empty	832*+2/932*-5+2*+
27	/	/	832*+2/932*-5+2*+
28	3	/	832*+2/932*-5+2*+3/

b. Tính giá trị biểu thức hậu tố trên theo bảng:

STT	Ký tự đang đọc	TT. Stack
1	8	8
2	3	8 3
3	2	8 3 2
4	*	8 6
5	+	14
6	2	14 2
7	/	7
8	9	7 9
9	3	7 9 3
10	2	7 9 3 2
11	*	7 9 6
12	-	7 3
13	5	7 3 5
14	+	7 8
15	2	7 8 2
16	*	7 16
17	+	23
18	3	23 3
19	/	7.6