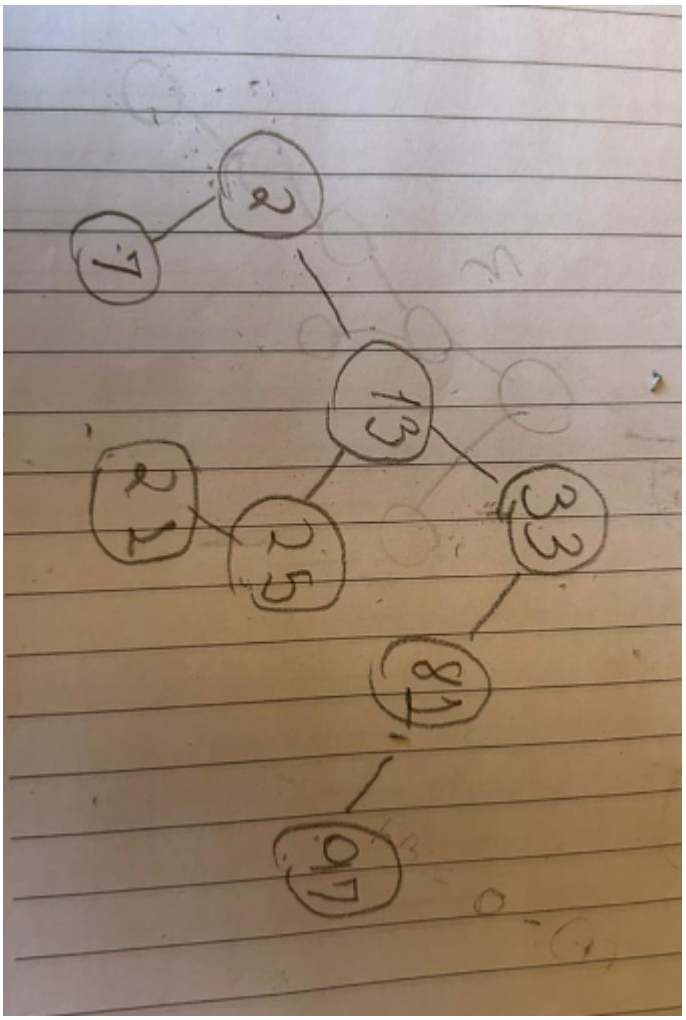
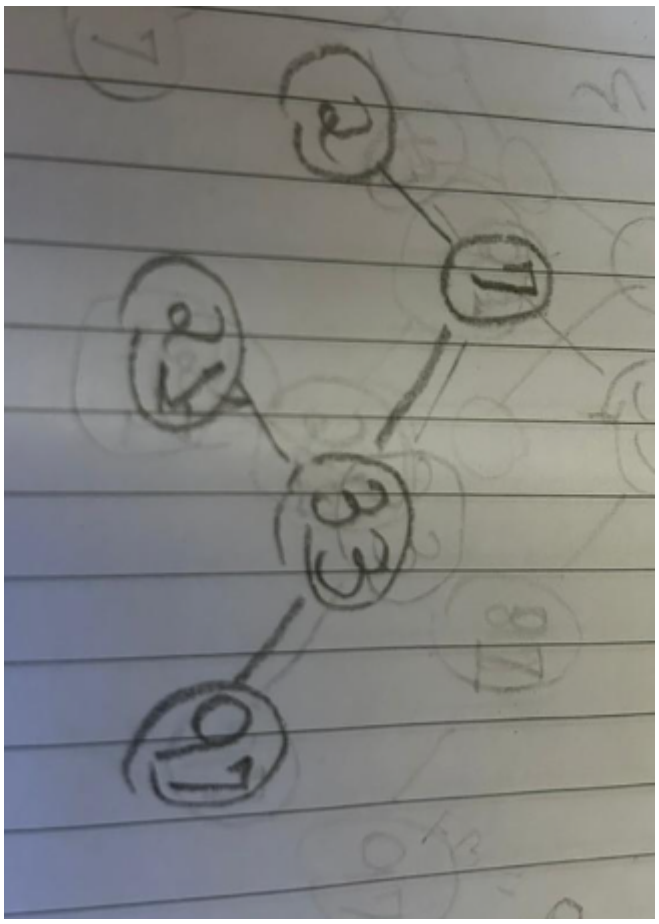
Questão 1

A)

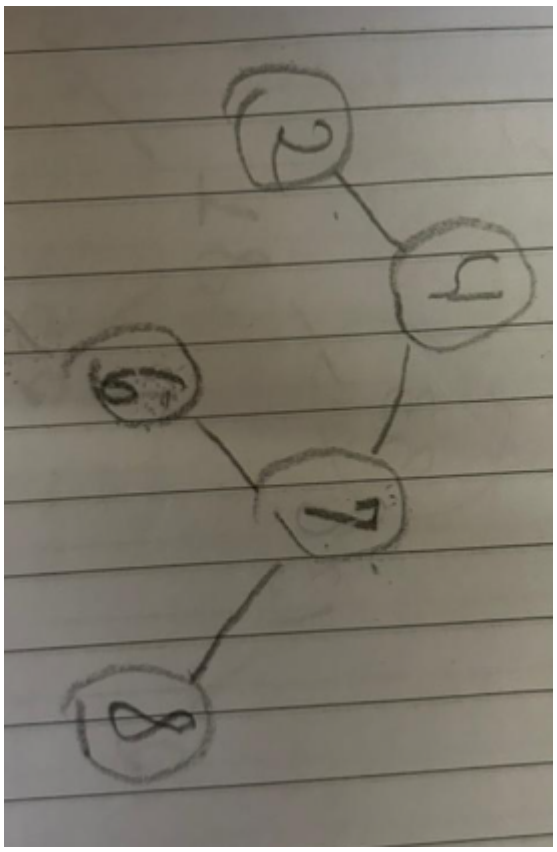


B)

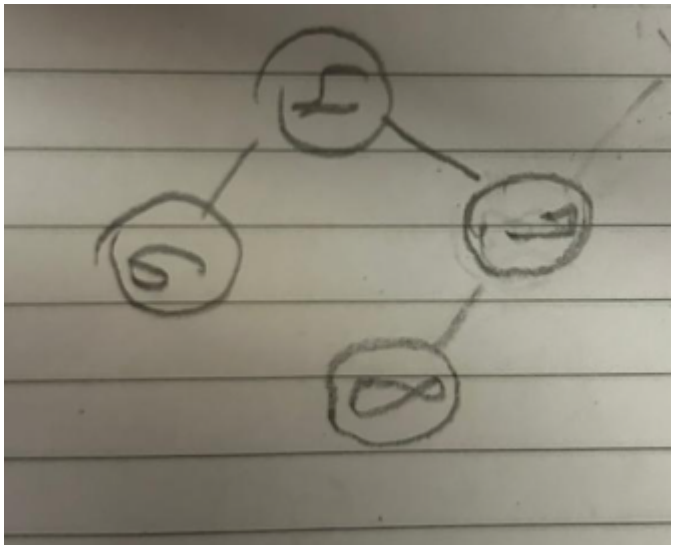


Questão 2

A)



B)



Questão 3 - Classes

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ABB
```

```
{
    class Inteiro
    {

        private int valor;
        public int Valor
        {
            get { return valor; }
            set { valor = value; }
        }

        public Inteiro(int valor)
        {
            this.valor = valor;
        }

        public Inteiro()
        {
            valor = 0;
        }

        public void imprimir()
        {
            Console.WriteLine("Valor -> " + valor);
        }
    }

    class No
    {

        private Inteiro item;
        public Inteiro Item
        {
            get { return item; }
            set { item = value; }
        }

        private No esquerda;
        public No Esquerda
        {
            get { return esquerda; }
            set { esquerda = value; }
        }
```

```
    private No direita;
    public No Direita
    {
        get { return direita; }
        set { direita = value; }
    }

    public No()
    {

        item = new Inteiro();
        esquerda = null;
        direita = null;
    }

    public No(Inteiro registro)
    {

        item = registro;
        esquerda = null;
        direita = null;
    }
}

class ABB
{

    private No raiz;

    public ABB()
    {

        raiz = null;
    }

    public Inteiro pesquisar(int chave)
    {
        return pesquisar(this.raiz, chave);
    }

    private Inteiro pesquisar(No raizSubarvore, int chave)
    {

        if (raizSubarvore == null)
```

```
            return null;
        else if (chave == raizSubarvore.Item.Valor)
            return raizSubarvore.Item;
        else if (chave > raizSubarvore.Item.Valor)
            return pesquisar(raizSubarvore.Direita, chave);
        else
            return pesquisar(raizSubarvore.Esquerda, chave);
    }

    public void inserir(Inteiro novo)
    {
        this.raiz = inserir(this.raiz, novo);
    }


    private No inserir(No raizSubarvore, Inteiro novo)
    {

        if (raizSubarvore == null)
            raizSubarvore = new No(novo);
        else if (novo.Valor == raizSubarvore.Item.Valor)
            throw new Exception("Não foi possível inserir o item na árvore: chave já
inseriada anteriormente!");
        else if (novo.Valor < raizSubarvore.Item.Valor)
            raizSubarvore.Esquerda = inserir(raizSubarvore.Esquerda, novo);
        else
            raizSubarvore.Direita = inserir(raizSubarvore.Direita, novo);

        return raizSubarvore;
    }

    public void remover(int chaveRemover)
    {
        this.raiz = remover(this.raiz, chaveRemover);
    }

    private No remover(No raizSubarvore, int chaveRemover)
    {

        if (raizSubarvore == null)
            throw new Exception("Não foi possível remover o item da árvore: chave
não encontrada!");
        else if (chaveRemover == raizSubarvore.Item.Valor)
        {
```

```
            if (raizSubarvore.Esquerda == null)
                raizSubarvore = raizSubarvore.Direita;
            else if (raizSubarvore.Direita == null)
                raizSubarvore = raizSubarvore.Esquerda;
            else
                raizSubarvore.Esquerda = antecessor(raizSubarvore,
raizSubarvore.Esquerda);
        }
        else if (chaveRemover > raizSubarvore.Item.Valor)
            raizSubarvore.Direita = remover(raizSubarvore.Direita, chaveRemover);
        else
            raizSubarvore.Esquerda = remover(raizSubarvore.Esquerda,
chaveRemover);

        return raizSubarvore;
    }

    private No antecessor(No noRetirar, No raizSubarvore)
    {

        if (raizSubarvore.Direita != null)
            raizSubarvore.Direita = antecessor(noRetirar, raizSubarvore.Direita);
        else
        {
            noRetirar.Item = raizSubarvore.Item;
            raizSubarvore = raizSubarvore.Esquerda;
        }

        return raizSubarvore;
    }

    public void caminhamentoPreOrdem()
    {
        caminhamentoPreOrdem(this.raiz);
    }

    private void caminhamentoPreOrdem(No raizSubarvore)
    {

        if (raizSubarvore != null)
        {
            raizSubarvore.Item.imprimir();
            caminhamentoPreOrdem(raizSubarvore.Esquerda);
            caminhamentoPreOrdem(raizSubarvore.Direita);
```

```
    }
}
public void caminhamentoPosOrdem()
{
    caminhamentoPosOrdem(this.raiz);
}

private void caminhamentoPosOrdem(No raizSubarvore)
{

    if (raizSubarvore != null)
    {
        caminhamentoPosOrdem(raizSubarvore.Esquerda);
        caminhamentoPosOrdem(raizSubarvore.Direita);
        raizSubarvore.Item.imprimir();
    }
}
public void caminhamentoEmOrdem()
{
    caminhamentoEmOrdem(this.raiz);
}

private void caminhamentoEmOrdem(No raizSubarvore)
{

    if (raizSubarvore != null)
    {
        caminhamentoEmOrdem(raizSubarvore.Esquerda);
        raizSubarvore.Item.imprimir();
        caminhamentoEmOrdem(raizSubarvore.Direita);
    }
}
public Inteiro pesquisarMaior()
{
    return pesquisarMaior(this.raiz);
}
private Inteiro pesquisarMaior(No maior)
{
    return maior.Direita == null ? maior.Item : pesquisarMaior(maior.Direita);
}
public Inteiro pesquisarMenor()
{
    return pesquisarMenor(this.raiz);
}
```

```
        private Inteiro pesquisarMenor(No maior)
        {
            return maior.Esquerda == null ? maior.Item :
pesquisarMaior(maior.Esquerda);
        }

    }
}
```

Questão 3 - Main

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lista15Questão3
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int escolha, numero;
            Inteiro num;
            ABB arvore = new ABB();
            do
            {
                Console.WriteLine("1- Inserir um número na árvore AVL\r\n2- Remover um
número da árvore AVL\r\n3- Pesquisar um número na árvore AVL\r\n4- Mostrar
todos os elementos da árvore AVL, usando o caminhamento central\r\n5- Mostrar
todos os elementos da árvore AVL, usando o caminhamento pós-ordem.\r\n6-
Mostrar todos os elementos da árvore AVL, usando o caminhamento
pré-ordem.\r\n7- Sair");
                escolha = int.Parse(Console.ReadLine());
                switch (escolha)
                {
                    case 1:
                        Console.WriteLine("Insira um número: ");
                        numero = int.Parse(Console.ReadLine());
                        num = new Inteiro(numero);
                        arvore.inserir(num);
                        break;
```

```
                case 2:
                    Console.WriteLine("Insira um número: ");
                    numero = int.Parse(Console.ReadLine());
                    arvore.remover(numero);
                    break;
                case 3:
                    Console.WriteLine("Insira um número: ");
                    numero = int.Parse(Console.ReadLine());
                    arvore.pesquisar(numero);
                    break;
                case 4:
                    arvore.caminhamentoEmOrdem();
                    break;
                case 5:
                    arvore.caminhamentoPosOrdem();
                    break;
                case 6:
                    arvore.caminhamentoPreOrdem();
                    break;
                default:
                    break;
            }
        } while (escolha != 7);
    }
  }
}
```