Questão 1

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



namespace Lista_14_Questão_1
 {
        class Program
        {
        static void Main(string[] args)
        {
        Console.WriteLine("Insira a quantidade de elementos do vetor: ");
        int size = int.Parse(Console.ReadLine());
        int[] vet = new int[size];
        for(int i = 0; i < size; i++)
        {
                Console.WriteLine("Insira um número do vetor: ");
                int num = int.Parse(Console.ReadLine());
                vet[i] = num;
        }
        Console.WriteLine("Digite o elemento procurado: ");
        int busca = int.Parse(Console.ReadLine());
        Console.WriteLine("Resposta: "+Pesquisa(vet, busca));
        Console.ReadLine();
        }
        public static bool Pesquisa(int[] vet, int busca)
        {
        return Pesquisa(vet, busca, 0, (vet.Length - 1));
        }
        private static bool Pesquisa(int[] vet, int busca, int esq, int dir)
        {
        Boolean resp;
        int meio = (esq + dir) / 2;
        if(esq > dir)
        {
                resp = false;
        }
        else if(busca == vet[meio])
        {
```

```
                resp = true;
        }
        else if(busca > vet[meio])
        {
                resp = Pesquisa(vet, busca, meio + 1, dir);
        }
        else
        {
                resp = Pesquisa(vet, busca, esq, meio - 1);
        }
        return resp;
        }
        }
}
```

Questão 2 - 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 2 - Main

```
 using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;
```

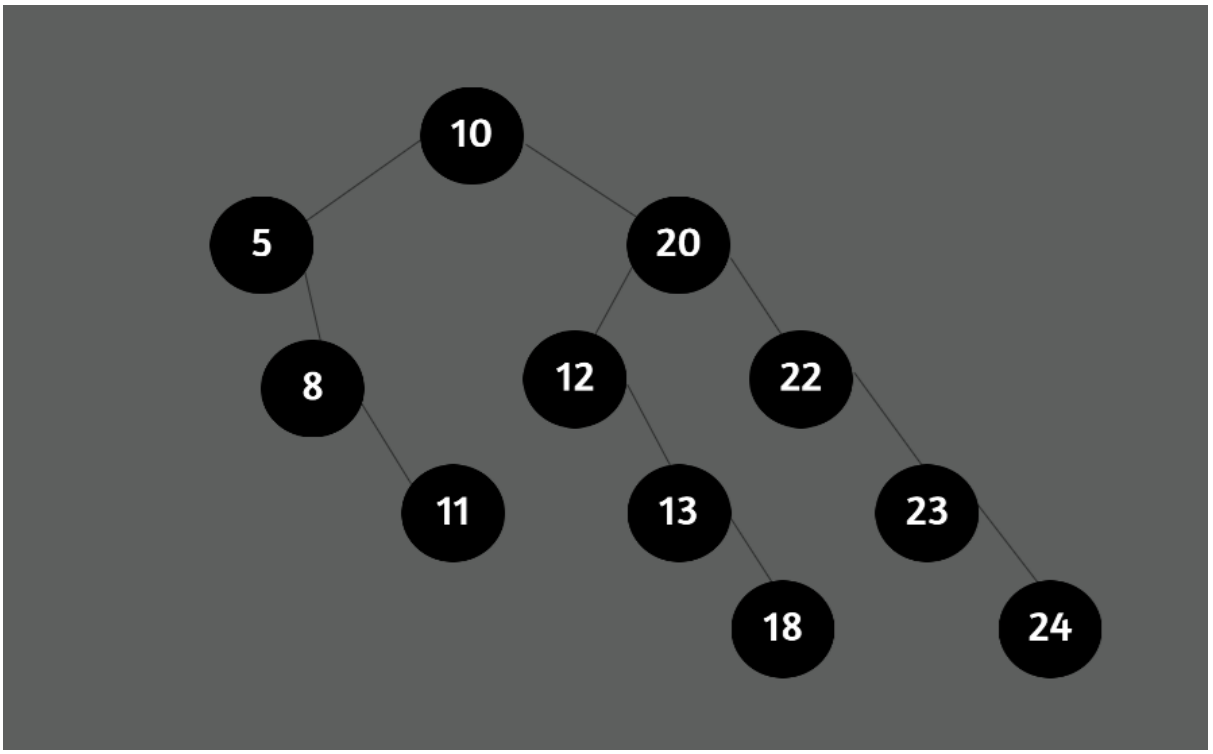
```
using System.Threading.Tasks;


namespace ABB
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int escolha, num;
            Inteiro item;
            ABB arvore = new ABB();
            do
            {
                Console.WriteLine("1- Inserir um número na árvore binária de busca\n" +
                    "2 - Remover um número da árvore binária de busca\n" +
                    "3 - Pesquisar um número na árvore binária de busca\n" +
                    "4 - Mostrar o maior elemento da árvore binária de busca\n" +
                    "5 - Mostrar o menor elemento da árvore de pesquisa de busca\n" +
                    "6 - Mostrar todos os elementos da árvore, usando o caminhamento
central\n" +
                    "7 - Mostrar todos os elementos da árvore, usando o caminhamento pós
- ordem.\n" +
                    "8 - Mostrar todos os elementos da árvore, usando o caminhamento pré
- ordem.\n" +
                    "9 - Sair");
                escolha = int.Parse(Console.ReadLine());
```

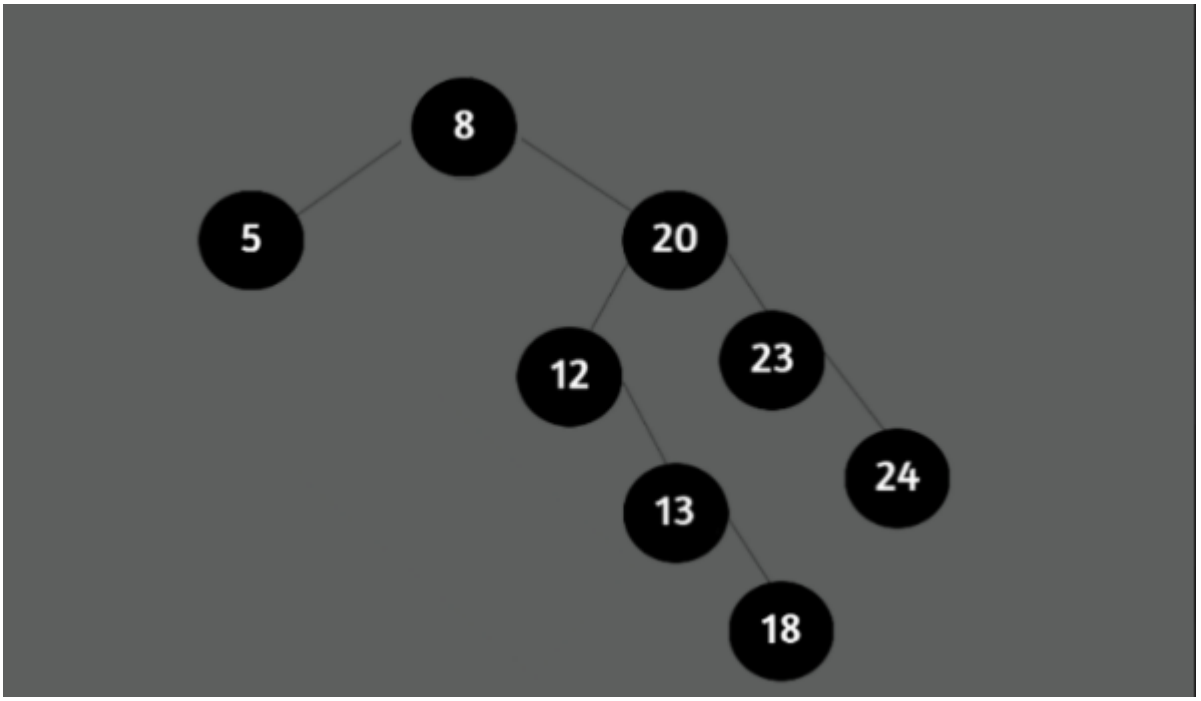
```
            switch (escolha)
            {
                case 1:
                    Console.WriteLine("Insira um número");
                    num = int.Parse(Console.ReadLine());
                    item = new Inteiro(num);
                    arvore.inserir(item);
                    break;
                case 2:
                    Console.WriteLine("Insira um número");
                    num = int.Parse(Console.ReadLine());
                    arvore.remover(num);
                    break;
                case 3:
                    Console.WriteLine("Insira um número");
                    num = int.Parse(Console.ReadLine());
                    Console.WriteLine(arvore.pesquisar(num).Valor + " encontrado com
sucesso.");
                    break;
                case 4:
                    Console.WriteLine(arvore.pesquisarMaior().Valor + " é o maior valor
da árvore");
                    break;
                case 5:
                    Console.WriteLine(arvore.pesquisarMenor().Valor + " é o menor valor
da árvore");
```

```
                    break;
                case 6:
                    arvore.caminhamentoEmOrdem();
                    break;
                case 7:
                    arvore.caminhamentoPosOrdem();
                    break;
                case 8:
                    arvore.caminhamentoPreOrdem();
                    break;
                default:
                    break;
            }
        } while (escolha != 9);
        Console.ReadLine();
    }
  }
}
```

Questão 3 - A



Questão 3 - B



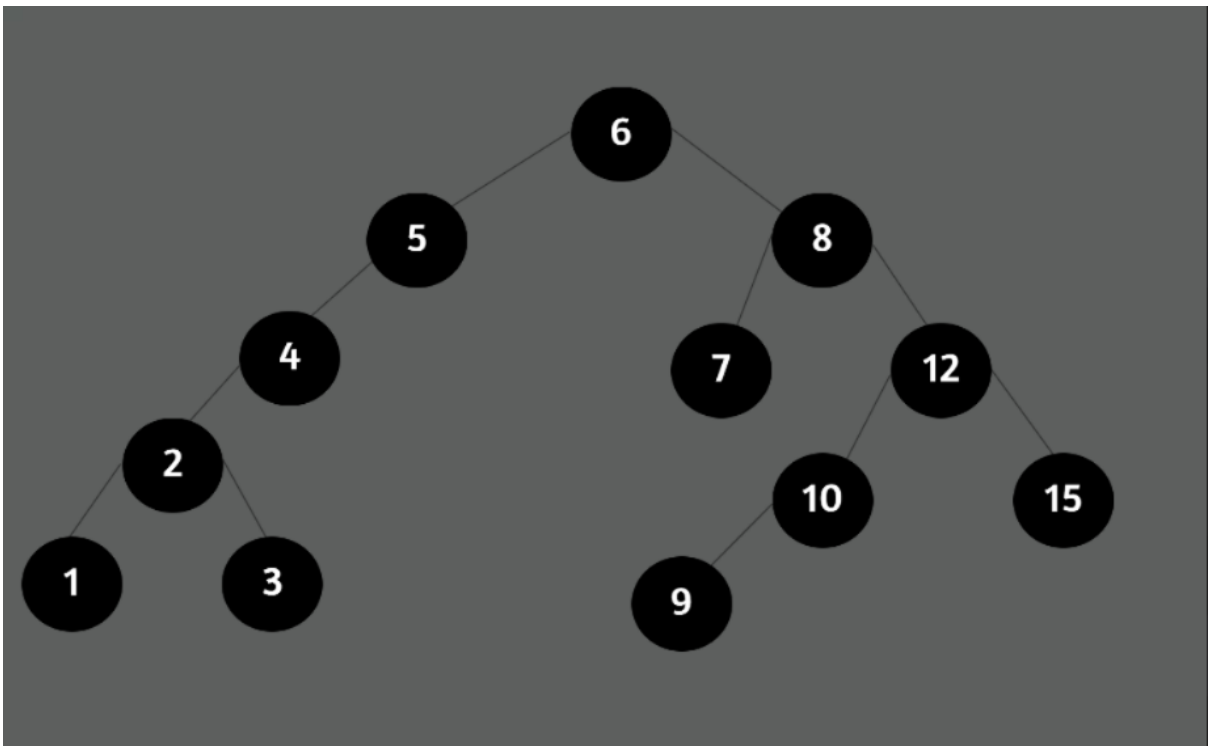
Questão 3 - C

Pré-Ordem: 10, 5, 8, 20, 12, 11, 13, 18, 22, 23, 24

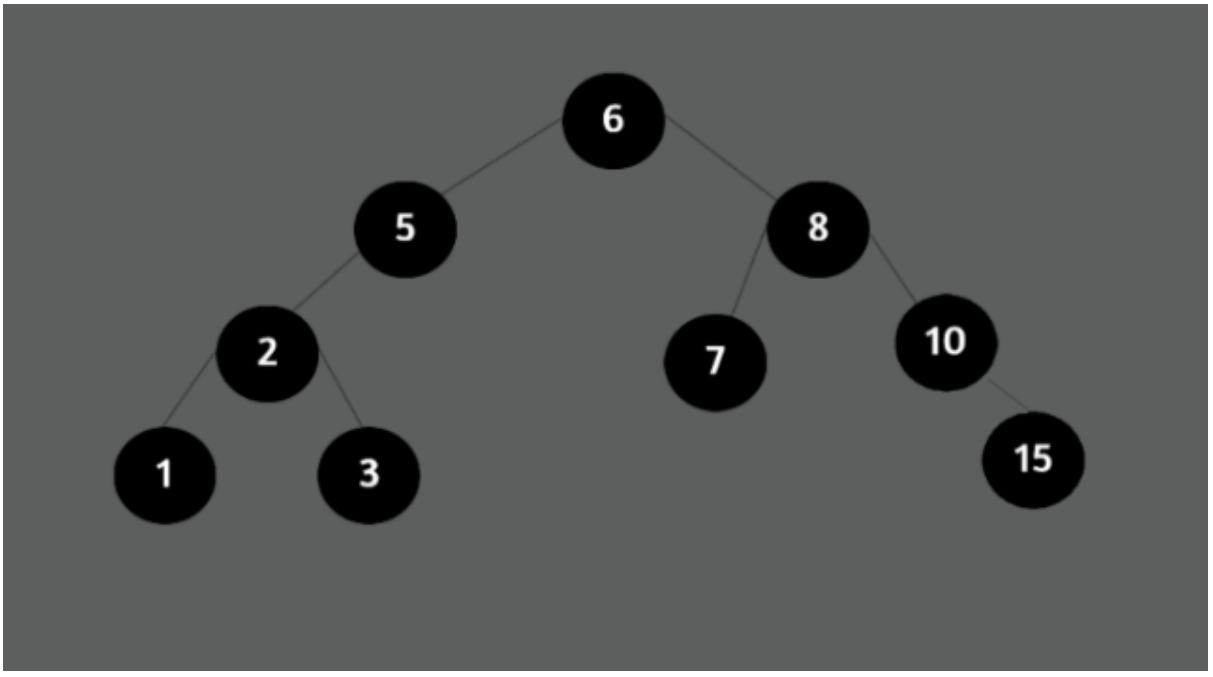
Pós-Ordem: 8, 5, 13, 18, 11, 12, 24, 23, 22, 20, 10

Central: 5, 8, 10, 11, 12, 13, 18, 20, 22, 23, 24

Questão 4 - A



Questão 4 - B



Questão 4 - C

Pré-Ordem: 6, 5, 4, 1, 3, 8, 7, 12, 10, 9, 15

Pós-Ordem: 3, 1, 4, 7, 9, 10, 15, 12, 8, 5, 6

Central: 1, 3, 4, 5, 6, 7, 8, 9, 10, 12, 15