Fibonacci with memoization
Caso repetir a questao da prova do ano passado, so fazer fib com cache: Array global de long pra salvar o cache -> instanciar o array global na main -> fazer o fibonacci: long fib n(int n) se n <= 1 retorna n; \n checar se o cache na pos n != 0, se sim retorna o cache[n], long ans = fib(n-2) + fib(n-1); \n cache[n] = ans; \n return ans;
Reverse linked list- Node reverseList(Node node){ \n if(node == null || node.next == null) return node; \n Node p = reverseList(node.next); \n node.next.next = node; \n node.next = null; //remove the pointer \n return p; }//end
Sum of all bt nodes- Int sumBT(Node cur){ \nif(cur == null){\n return 0; }\nreturn cur.elemento + sumBT(cur.left) + sumBT(cur.right);}
Decimal to binary- solve(int dec){ if(dec == 0) return 0; else return(dec % 2 + 10 * solve(dec / 2) }

Classes Github

```
public class Fila {
private Celula primeiro;
private Celula ultimo;
public Fila() {primeiro = new Celula(); ultimo = primeiro;}
public void inserir(int x) {
ultimo.prox = new Celula(x);
ultimo = ultimo.prox;
}
public int remover() throws Exception {
if (primeiro == ultimo) {throw new Exception("Erro ao remover!");}
Celula tmp = primeiro;
primeiro = primeiro.prox;
int resp = primeiro.elemento;
tmp.prox = null;
tmp = null;
return resp;
}
}
public class Pilha {
private Celula topo;
public Pilha() {topo = null;}
public void inserir(int x){Celula tmp = new Celula(x); tmp.prox = topo; topo = tmp; tmp = null;}
public int remover() throws Exception {
if (topo == null) {throw new Exception("Erro ao remover!");}
int resp = topo.elemento;
Celula tmp = topo;
topo = topo.prox;
tmp.prox = null;
tmp = null;
return resp;
}
public int getSoma() { return getSoma(topo); }
private int getSoma(Celula i) {
                                int resp = 0;
if (i != null) {
resp += i.elemento + getSoma(i.prox);
}
return resp;
}
public int getMax() {
int max = topo.elemento;
                for (Celula i = topo.prox; i != null; i = i.prox) { if (i.elemento > max) max = i.elemento; }
return max;
}
}
class ListaDupla {private CelulaDupla primeiro;private CelulaDupla ultimo;
                public ListaDupla() { primeiro = new CelulaDupla(); ultimo = primeiro;}
public void inserirInicio(int x) {
CelulaDupla tmp = new CelulaDupla(x);
    tmp.ant = primeiro;
    tmp.prox = primeiro.prox;
    primeiro.prox = tmp;
    if(primeiro == ultimo){ ultimo = tmp;
    Else tmp.prox.ant = tmp;
    tmp = null;
}
```

```
public void inserirFim(int x) {ultimo.prox = new CelulaDupla(x); ultimo.prox.ant = ultimo; ultimo = ultimo.prox; }
public int removerInicio() throws Exception {
if (primeiro == ultimo) throw new Exception("Erro ao remover (vazia)!");
                CelulaDupla tmp = primeiro;
                primeiro = primeiro.prox;
                int resp = primeiro.elemento;
                tmp.prox = primeiro.ant = null;
                tmp = null;
return resp;
}
public int removerFim() throws Exception {
if (primeiro == ultimo) { throw new Exception("Erro ao remover (vazia)!"); }
    int resp = ultimo.elemento;
    ultimo = ultimo.ant;
    ultimo.prox.ant = null; ultimo.prox = null; return resp; }
  public void inserir(int x, int pos) throws Exception {
    int tamanho = tamanho();
    if(pos < 0 || pos > tamanho) throw new Exception("Erro ao inserir posicao (" + pos + " / tamanho = " + tamanho + ") invalida!");
     else if (pos == 0) inserirInicio(x);
     else if (pos == tamanho) inserirFim(x);
     else {
       CelulaDupla i = primeiro;
       for(int j = 0; j < pos; j++, i = i.prox);
       CelulaDupla tmp = new CelulaDupla(x);
       tmp.ant = i;
       tmp.prox = i.prox;
       tmp.ant.prox = tmp.prox.ant = tmp;
       tmp = i = null;
    }
  }
public int remover(int pos) throws Exception {
    int resp; int tamanho = tamanho();
     if (primeiro == ultimo) throw new Exception("Erro ao remover (vazia)!");
     else if(pos < 0 || pos >= tamanho){ throw new Exception("Erro ao remover (posicao " + pos + " / " + tamanho + " invalida!");
     else if (pos == 0) resp = removerInicio();
     else if (pos == tamanho - 1) resp = removerFim();
     else {
       CelulaDupla i = primeiro.prox;
       for(int j = 0; j < pos; j++, i = i.prox);
       i.ant.prox = i.prox;
       i.prox.ant = i.ant;
       resp = i.elemento;
       i.prox = i.ant = null;
       i = null;
    } return resp;}}
```

BT

```
private No inserir(int x, No i) throws Exception {
    if (i == null) i = new No(x);
    else if (x < i.elemento) i.esq = inserir(x, i.esq);
    else if (x > i.elemento) i.dir = inserir(x, i.dir);
    else throw new Exception("Erro ao inserir!");
    return i;
}
private No remover(int x, No i) throws Exception {
if (i == null) throw new Exception("Erro ao remover!");
      else if (x < i.elemento) i.esq = remover(x, i.esq);
      else if (x > i.elemento) i.dir = remover(x, i.dir);
      else if (i.dir == null) i = i.esq;
      else if (i.esq == null) i = i.dir;
      else i.esq = maiorEsq(i, i.esq);
      return i;
}
```