```
Alunos: Bruno Menegotto
        Bruno Seixas
Curso: Engenharia de Computação
Disciplina: Estrutura de Dados I
Prof. Dr. Marcio Augusto de Souza

Lista Encadeada: Implementada com Vetor:

#include<iostream>
using namespace std;
const int TAM = 10;

int lista[TAM];
int comprimento = 0;

void inserelista(int valor)
{
    int i;
    if (comprimento == TAM)
    {
        cout << "\nLista Cheia";
        return;
    }
    for (i = comprimento; i > 0 && valor < lista[i-1]; i--)
        lista[i] = lista[i-1];
        lista[i] = valor;
        comprimento ++;
        cout << "\nValor " << valor << " inserido";
}
void removeValor(int valor)
{
    int i, j;
    for(i = 0; i < comprimento; i++)
    {
        if(valor == lista[i])
        {
            for(j = i; j < comprimento -1; j++)
                lista[j] = lista[j+1];
                cout << "\nRemovido o valor " << valor;
                comprimento --;
                return;
        }
    }
    cout << "\nValor nao encontrado";
}
void removeLista(int posicao)
{
    int i, j;
    for(i = 0; i < comprimento; i++)
    {
        if (posicao == i)
        {
            for(j = i; j < comprimento -1; j++)
                lista[j] = lista[j+1];
                cout << "\nRemovido o valor da " << posicao+1 << " posicao";
                comprimento --;
                return;
        }
    }
    cout << "\nPosicao " << posicao << " nao encontrado";
}
void recuperaLista(int posicao)
{
    if (posicao < 0 || posicao > comprimento)
    {
        cout << "\nPosicao " << posicao << " Invalida";
```

```
        return;
    }
    cout << "\nElemento na posicao " << posicao << " foi recuperado seu valor eh " << lista[posicao];
}
void imprimir()
{
    int i;
    if (comprimento == 0)
    {
        cout << "\nLista Vazia";
        return;
    }
    cout << "\n";
    for(i = 0; i < comprimento; i++)
        cout << lista[i] << " ";
}

int main()
{
    inserelista(2);
    inserelista(5);
    inserelista(6);
    inserelista(9);
    inserelista(10);
    inserelista(15);
    imprimir();
    removeValor(9);
    removeValor(2);
    inserelista(1);
    imprimir();
    removeLista(0);
    imprimir();
    recuperaLista(0);
    imprimir();

}
```

Lista Encadeada: Implementada com Alocação Dinamica

```
#include<iostream>
using namespace std;

struct no
{
    int dado;
    struct no *prox;
};

struct no *lista = NULL;
void insereLista(int valor)
{
    struct no *atual;
    struct no *anterior;
    struct no *novo;
    novo = new(struct no);
    novo -> dado = valor;
    atual = lista;
    while (atual != NULL && atual -> dado < valor)
    {
        anterior = atual;
        atual = atual -> prox;
    }
    if (atual == lista)
    {
        lista = novo;
        novo -> prox = atual;
```

```
133     }
134     else
135     {
136         anterior -> prox = novo;
137         novo -> prox = atual;
138     }
139     cout << "\nElemento " << valor << " inserido com sucesso";
140 }
141 void removeLista(int posicao)
142 {
143     struct no *atual;
144     struct no *aux;
145     int cont = 1;
146     if(lista == NULL)
147     {
148         cout << "\nLista Vazia";
149         return;
150     }
151     atual = lista;
152     while(atual != NULL && cont <= posicao)
153     {
154         atual = atual -> prox;
155         cont++;
156     }
157     if(posicao < 0 || atual == NULL)
158     {
159         cout << "\nPosicao " << posicao << " para remocao nao existe";
160         return;
161     }
162     if(posicao == 0)
163     {
164         aux = atual;
165         lista = lista -> prox;
166     }
167     else
168     {
169         aux = atual -> prox;
170         atual -> prox = aux -> prox;
171     }
172     delete(aux);
173     cout << "\nElemento da posicao " << posicao << " removido";
174 }
175 void recuperaLista(int posicao)
176 {
177     struct no *atual;
178     int cont = 0;
179     atual = lista;
180     while(atual != NULL && cont <= posicao)
181     {
182         atual = atual -> prox;
183         cont ++;
184     }
185     if(posicao < 0 || atual == NULL)
186     {
187         cout << "\nPosicao " << posicao << " para a recuperacao nao existe";
188         return;
189     }
190     cout << "\nElemento na posicao " << posicao << " foi recuperado seu valor eh " << atual -> dado;
191 }
192 void removeValor(int valor)
193 {
194     struct no *atual;
195     struct no *aux;
196     int cont = 0;
197     if(lista == NULL)
198     {
```

```
199         cout << "\nLista Vazia";
200         return;
201     }
202     atual = lista;
203     while(atual != NULL && cont <= valor)
204     {
205         if(atual -> dado == valor)
206         {
207             if(atual == lista)
208             {
209                 lista = lista -> prox;
210             }
211             else
212             {
213                 aux -> prox = atual -> prox;
214             }
215             delete(aux);
216             cout << "\nElemento " << valor << " removido";
217             return;
218         }
219         aux = atual;
220         atual = atual -> prox;
221         cont++;
222     }
223     cout << "\nElemento " << valor << " para a nao existe";
224 
225 }
226 void imprimir()
227 {
228     struct no *aux;
229     if(lista == NULL)
230     {
231         cout << "\nLista Vazia";
232         return;
233     }
234     aux = lista;
235     cout << "\nLista eh: ";
236     while (aux!=NULL)
237     {
238         cout << aux -> dado << " ";
239         aux = aux -> prox;
240     }
241 }
242 int main()
243 {
244     insereLista(1);
245     insereLista(2);
246     insereLista(3);
247     insereLista(5);
248     insereLista(10);
249     insereLista(4);
250     imprimir();
251     removeLista(6);
252     imprimir();
253     recuperaLista(6);
254     imprimir();
255     removeValor(20);
256     imprimir();
257 }
258 
```