4 (30 pontos) Balanceado

Uma sequência de N elementos V_i ORDENADOS, com N par, é dita balanceada quando a soma do maior elemento com o menor elemento é igual a soma do segundo maior elemento com o segundo menor elemento, que por sua vez é igual a soma do terceiro maior elemento com o terceiro menor elemento, e assim por diante. Escreva uma função **RECURSIVA** em `C` que decida se o vetor está balanceado ou não. Esta função deverá retornar 1 quanto estiver balanceado e 0 quando não estiver. Implemente quaisquer funções auxiliares que achar necessário.

A função tem o seguinte protótipo (e não pode ser modificado):

```
int balanceada(int *vetor, int inicio, int fim)
```

Restrições:
- $2 \leq N \leq 1000$
- $-1000 \leq V_i \leq 1000$

Exemplo de vetor balanceado:

```
1 3 7 11 15 17
```

Exemplo de vetor não balanceado:

```
3 3 5 8 9 10 11 27 32 41 45 51
```

4.1 Escreva a sua resposta no espaço abaixo:

Para resolver esta questão, podemos abordar o problema de duas maneiras semelhantes: uma abordando manipulação de ponteiros diretamente no vetor, e; outra mais simples manipulando os valores de início e fim, sendo esta a solução esperada.

Para ambas as versões é necessária a criação de um *wrapper* (invólucro) da função balanceada recursiva para carregar o valor da soma esparada em cada etapa.

A Solução Esperada, abaixo, cria utiliza a função definida no problema apenas para calcular a soma inicial e chamar a função recursiva `balanceadaR` já incrementando a variável `inicio` e decrementando a variável `fim`. A função `balanceadaR` possui como critério de parada o momento que o lado mais esquerdo, `l`, do vetor ultrapassa o lado mais a direita, `r`. Caso a recursão chegue nesse ponto é porque todas as somas anteriores foram iguais à esperada. O próximo passo é fazer a soma local e caso ela seja diferente do esperado já finaliza a função devolvendo 0 (o valor esperado para caso o vetor não esteja balanceado), e; caso contrário a função se chama recursivamente incrementando `l` e decrementando `r`.

```
 1  int balanceadaR(int *v,int l,int r,int sa)
 2  {
 3    if(l>r)
 4      return 1;
 5    int s=v[l]+v[r];
 6    if(s!=sa)
 7      return 0;
 8    return balanceadaR(v,l+1,r-1,sa);
 9  }
10  int balanceada(int *vetor,int inicio,int fim)
11  {
12    return balanceadaR(vetor,inicio+1,fim-1,vetor[inicio]+vetor[fim]);
13  }
```

Uma outra abordagem para este problema, é a utilização apenas do ponteiro de início do vetor e a quantidade de elementos, em intervalo fechado, presentes no vetor. Esta solução é equivalente à primeira e o leitor pode, trivialmente, entender os conceitos aplicados.

```
 1  int balanceadaR(int *v, int r,int sa)
 2  {
 3    if(v>&v[r])
 4      return 1;
 5    int s=v[0]+v[r];
 6    if(s!=sa)
 7      return 0;
 8    return balanceadaR(&v[1],r-2,sa);
 9  }
10  int balanceada(int *vetor, int inicio, int fim)
11  {
12    return balanceadaR(&vetor[inicio+1],fim-2,vetor[inicio]+vetor[fim]);
13  }
```