Avaliação do horário de 14h – 16h

Leia com atenção as instruções a seguir:

- A avaliação é **individual** e **sem qualquer tipo de consulta**.
- A resposta de uma questão `i` deve, **obrigatoriamente**, estar entre o par de marcadores `//Qi` e `//-Qi`. Ou seja, a resposta da questão 1 deve estar entre `//Q1` e `//-Q1`, a resposta da questão 2 deve estar entre `//Q2` e `//-Q2`, e assim por diante. Caso sua solução para uma determinada questão tenha várias funções ou operações, todas elas devem estar entre os marcadores da questão. **Não remover, em hipótese alguma, tais marcadores de questões da sua prova!**
- **É sua responsabilidade enviar as respostas ao servidor** ao terminar de resolver a avaliação. Para isso, acesse o endereço `http://10.5.112.11:8080/alg2-2024-3-tvc1/` e, usando a janela de upload, envie o arquivo `main.cpp` com suas respostas para as questões da avaliação. Confira as suas respostas e, somente então, **confirme o envio**.

Questões:

1. (10 Pontos) Implemente a função `void q1()` com o que se pede a seguir:
   (a) Declare um vetor estático `vet` com 6 inteiros, inicializado com os valores {`10,20,30,40,50,60`}.
   (b) Declare um ponteiro para inteiro `p` e aponte o ponteiro para o ínicio do vetor.
   (c) Avance o ponteiro `p` para que ele aponte para o elemento `50`.
   (d) A partir da nova posição de `p`, retroceda uma posição.
   (e) Salve em uma variável inteira a diferença entre o endereço em `p` e o do primeiro elemento do vetor.
   (f) Imprima o valor calculado dessa diferença.
   (g) Agora, faça `p` apontar para o último elemento do vetor.
   (h) Volte `p` para que ele aponte novamente para o primeiro elemento do vetor.
   (i) Imprima o valor do elemento apontado por `p`.
   (j) Usando a notação `&vet[...]`, aponte `p` para o segundo elemento do vetor.
   (k) Usando `p` (sem alterar seu valor) e aritmética de ponteiros, imprima o terceiro elemento do vetor.

```
void q1();
```

2. (10 Pontos) Implemente a função `int* pares(int vet[], int n, int *m, int *t)` que recebe como parâmetros um vetor `vet` de elementos inteiros e seu tamanho `n`. A função deve calcular o maior valor par do vetor e atribuir este resultado à variável `m`, passada por referência como parâmetro. Se não houver nenhum valor par no vetor, o valor `-1` deve ser atribuído à variável `m`. Além disso, a função deve alocar dinamicamente um novo vetor e preencher este novo vetor com todos os elementos pares no vetor. O novo vetor deve ser alocado com o número adequado de posições. Se não houver nenhum valor par no vetor, a função deve retornar `NULL`. Ao final, o novo vetor criado na função deve ser retornado e o tamanho deste novo vetor deve ser atribuído à variável `t`, passada por referência como parâmetro.

```
int* pares(int vet[], int n, float *m, int *t);
```

3. (10 Pontos) Implemente a função **recursiva** `void imprimeBinario(int n)` para imprimir a representação binária do número inteiro `n`.

```
void imprimeBinario(int n);
```

4. (10 Pontos) Implemente a função **recursiva** `bool ehOrdenado(int vet[], int n)` que recebe como parâmetros um vetor de elementos inteiros `vet` e seu tamanho `n`. A função deve retornar `true` se o vetor estiver estritamente em ordem crescente (cada elemento é menor que o seguinte) e `false` caso contrário.

```
bool ehOrdenado(int vet[], int n);
```