INSTITUTO FEDERAL
Rio Grande do Norte
Campus Natal-Central

DIATINF – Diretoria Acadêmica de Gestão e Tecnologia da Informação

Linguagem de programação C++ – Curso de férias

Exercícios 05 – C++ Alocação dinâmica de memória

Professor Jorgiano Marcio Bruno Vidal

1. Sobre o programa a seguir:

```
#include <iostream>

int main() {
    int x, y;
    std::cin >> x >> y;
    int *px = &x;
    *px = x + y;
    px = &y;
    y = y + 1;
    *px = *px + 2;
    std::cout << x << " " << y << " " << *px << std::endl;
    return 0;
}
```

   (a) Escreva a saída do programa para as entradas 10 e 20.

   (b) Faça um desenho da memória a cada passo.

2. Sobre o programa a seguir:

```
#include <iostream>

int main() {
    int x, y, *p = new int;
    std::cin >> x >> y >> *p;
    int *q = p;
    p = &x;
    *p = *p + *q;
    *q = *q + 1;
    std::cout << x + y << " " << *p << " " << *q << std::endl;
    return 0;
}
```

   (a) Escreva a saída do programa para as entradas 2 e 20.

   (b) Escreva o estado da memória a cada comando executado do programa.

   (c) Durante a execução do programa um espaço de memória foi alocado com comando `new`. Escreva o comando para liberar o espaço solicitado antes do término do programa.

3. O programa seguir lê um sequencia de números inteiros e depois mostra os números, na ordem em que foram lidos.

```
#include <iostream>

int main() {
    int x, *a = new int[100], capacidade = 100, tamanho = 0;
    while (std::cin >> x, x != -1) {
        if (tamanho == capacidade) {
            /* Precisa aumentar a capacidade do array*/
        }
        a[tamanho] = x;
        tamanho = tamanho + 1;
    }
    for (int i = 0; i < tamanho; ++i) {
        std::cout << a[i] << " ";
    }
    std::cout << std::endl;
    delete[] a;
    return 0;
}
```

Ao alcançar a capacidade do *array* é necessário solicitar mais memórra para continuar lendos os números. Escreva o trecho de código para aumentar a memória alocada para o *array* a.

4. O programa a seguir lê um *array* de números inteiros e deve calcular um novo *array* com os números primos pertencentes a a.

```
#include <iostream>

std::tuple<unsigned int, int *> calcula_primos(const int *a, unsigned int tamanho) {
    int tam_primos, *primos;
    /* TODO */
    return {tam_primos, primos};
}

int main() {
    int tamanho;
    std::cin >> tamanho;
    int *a = new int[tamanho];
    for (int i = 0; i < tamanho; ++i)
        std::cin >> a[i];
    std::cout << "Números lidos: ";
    for (int i = 0; i < tamanho; ++i) {
        std::cout << a[i] << " ";
    }
    std::cout << std::endl;
    auto [tam_primos, primos] = calcula_primos(a, tamanho);
    std::cout << "São primos: ";
    for (int i = 0; i < tam_primos; ++i) {
        std::cout << primos[i] << " ";
    }
    std::cout << std::endl;
    delete[] a;
    delete[] primos;
    return 0;
}
```

Entenda o código e implemente a função que retorna o *array* com os números primos.

5. O programa a seguir lê um número inteiro n ($1 \leq n \leq 10^18$) e deve calcular e mostrar os fatores primos de n.

```
#include <iostream>

std::tuple<unsigned int, int *> fatorar(long long n) {
    int tam_fatores, *fatores;
    /* TODO */
    return {tam_fatores, fatores};
}

void print_fatores(const int *a, int t) {
    std::cout << a[0]; // Um número tem, pelo menos, 1 fator.
    for (int i = 1; i < t; ++i) {
        std::cout << " x " << a[i];
    }
}

int main() {
    long long n;
    std::cin >> n;
    auto [tam_fatores, fatores] = fatorar(n);
    std::cout << n << " = ";
    print_fatores(fatores, tam_fatores);
    std::cout << std::endl;
    delete[] fatores;
    return 0;
}
```

Implemente a função que calcula os fatores primos de n.