Instruções

1. A prova vale 100;
2. Você tem uma hora e 30 minutos para resolver as questões;
3. O acesso à Internet será então restabelecido para a submissão das respostas;
4. A solução do exercício 1 deve estar no arquivo *1.cpp*;
5. A solução do exercício 2 deve estar no arquivo *2.cpp*;
6. A solução do exercício 3 deve estar no arquivo *3.cpp*;
7. Comprima os arquivos fontes com o comando `tar -czvf pp3.tar.gz 1.cpp 2.cpp 3.cpp`
8. Envie o arquivo `pp3.tar.gz` para `gcom.tp.sub@gmail.com`;
9. O envio deve ser feito obrigatoriamente de seu e-mail institucional da UFV;
10. O assunto (*subject*) do e-mail deve ser apenas seu número de matrícula (sem o ES);
11. Nenhum tipo de plágio será tolerado;
12. **A interpretação faz parte da avaliação.**

1. (30 Pontos) Faça um programa, que use operadores em bits, para verificar se um número (inteiro de 32 bits sem sinal) fornecido pela linha de comando é par ou ímpar. Se o número for par, seu programa deve imprimir 0 e se o número for ímpar, seu programa deve imprimir 1.

   Observação você não está autorizado a usar operadores aritméticos (-, +, *, / e %) e comandos de repetição.

   Exemplo:

```
g++ -o 1 1.cpp
./1 0
0
./1 1
1
./1 2
0
./1 123
1
```

2. (35 Pontos) Um aluno da disciplina de CAP CXII teve a tarefa de implementar o *RadixSort* na versão LSD (*Least Significant Digit*) para inteiros não negativos de 32 bits (utiliza o *RadixSort* começando pelo bit menos significativo). Após algumas dicas do professor, o aluno chegou a seguinte implementação:

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

void lsdRadixSort(unsigned int *v, int n) {
    unsigned int *vaux, *paux, mask;
    int i, pos, flag;
    // vetor auxiliar
    vaux = new unsigned int[n];

    for(mask = 1 << 31, flag = 1; mask; mask >>= 1) {
        for(i = pos = 0; i < n; i++)
            if(!(v[i] & mask))
                vaux[pos++] = v[i];
        for(i = 0; pos < n; i++)
            if(v[i] & mask)
                vaux[pos++] = v[i];
        paux = v;
        v = vaux;
        vaux = paux;
        flag = !flag;
    }

    if(!flag) {
        for(i = 0; i < n; i++)
            vaux[i] = v[i];
        delete []v;
        v = vaux;
    } else {
        delete []vaux;
    }
}

int main() {
    int n;
    int i;
    cin >> n;
    unsigned int *x = new unsigned int[n];
    for (i = 0; i < n; i++)
        cin >> x[i];
    lsdRadixSort(x, n);
    for (i = 0; i < n; i++)
        cout << x[i] << endl;
    delete []x;
    return 0;
}
```

Ao perceber que o código não funcionava, o aluno foi ao professor pedir ajuda. O professor disse ao aluno: "Bom, você fez praticamente tudo certo. Só está confundindo MSD e LSD. Se perceber isso, mudando poucos caracteres, seu código vai funcionar".

Siga a dica do professor de CAP CXII e conserte o código.

3. (35 Pontos) Seja S um conjunto com n elementos distintos (inteiros não negativos de 32 bits) e k um número (também um inteiro não negativo de 32 bits). Desenvolva um programa, baseado em backtracking **com poda**, para encontrar, dados S e k, o subconjunto de S cuja soma dos elementos seja máxima e menor ou igual a k. Para lhe ajudar, abaixo o código que vimos em sala para gerar combinações.

```
void imprimeCombinacoes(bool combs[], int begin, int n) {
    if (begin >= n) {
        for(int i=0;i<n;i++) {
            cout << combs[i];
        }
        cout << endl;
    } else {
        combs[begin] = 0;
        imprimeCombinacoes(combs, begin+1, n);
        combs[begin] = 1;
        imprimeCombinacoes(combs, begin+1, n);
    }
}
int main(int argc, char **argv) {
    int n = atoi(argv[1]);
    bool combs[n];
    imprimeCombinacoes(combs,0,n);
}
```

Seu programa deve primeiro ler k, então n e depois o valor de cada um dos elementos. A saída do programa deve ser o subconjunto encontrado, quando existir. Se houver mais de um conjunto, você pode imprimir qualquer um deles (e em qualquer ordem). Se S não possuir nenhum subconjunto com soma menor ou igual a k, seu programa não deve imprimir nada.

Exemplo:

```
g++ -o 3 3.cpp
./3
1
3
2 3 4

./3
1
5
1 2 3 4 5
1
```

Explicação: No primeiro exemplo, é impossível encontrar um subconjunto de $\{2, 3, 4\}$ que some 1 ou menos. No segundo, o subconjunto $\{1\}$ tem soma 1.

Observação: Você pode usar variáveis globais nesse exercício, mas com moderação!