

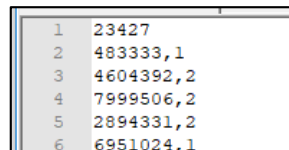


Lista Quarto Bimestre – Peso 3

Observações:

- A lista de exercícios pode ser feita por grupos de até 4 pessoas;
 - A entrega deve ser realizada via MOODLE até a data da prova;
 - A cada hora de atraso, o valor máximo possível de ser atingido pelo grupo diminuirá em 10%;
 - Portanto, se a entrega ocorrer com $0 < \text{hora} \leq 1$ de atraso, os alunos só poderão atingir 90% da nota integral da lista; com $1 < \text{hora} \leq 2$ de atraso, 80% e assim em diante;
 - Somente deverão ser entregues:
 - Relatório especificando a lógica utilizada e os resultados obtidos;
 - Código fonte, ou seja, os arquivos .c (e .h se existir);
 - Arquivos de texto com os resultados, quando aplicável.
 - Aos códigos fonte que não compilarem, será atribuída nota zero.
-

1. (1,0) Crie um algoritmo capaz de dividir um conjunto de dados em dois grupos de tal maneira que a diferença entre a soma dos elementos dos conjuntos seja a mínima possível. Aplique o algoritmo desenvolvido no arquivo 'numeros_particao.txt', que está na pasta da disciplina. Como resposta, o programa deve-se gerar um arquivo de texto como o exemplo seguinte:



1	23427
2	483333,1
3	4604392,2
4	7999506,2
5	2894331,2
6	6951024,1

Figura 1 - Exemplo do arquivo de resultados.

A primeira linha mostra a diferença entre a soma dos elementos dos conjuntos, ou seja:

$$\text{primeiraLinha} = \left| \sum \text{elementosConjunto1} - \sum \text{elementosConjunto2} \right|$$

Na segunda linha, é mostrado o primeiro valor do arquivo 'numeros_particao.txt' **vírgula** qual o conjunto ele pertence (1 ou 2), classificado de acordo com o algoritmo desenvolvido. O mesmo ocorre para as demais linhas até o último número presente no arquivo original. Note que a Figura 1 mostra apenas uma pequena parte (inicial) do arquivo que deve ser gerado como resultado. O valor da primeira linha nem a classificação dos 5 primeiros números nos conjuntos 1 ou 2 também não apresentam relação com o melhor resultado. Portanto, a figura é meramente ilustrativa!

Não é permitida o uso de bibliotecas que não sejam: *stdio.h*, *stdlib.h*, *string.h*, *stdbool.h* e *math.h*. A pontuação do exercício ocorrerá da seguinte maneira:

1. Somente receberão pontuação as implementações que estiverem sem erros;
2. O grupo que apresentar como resultado uma diferença entre a soma dos elementos dos conjuntos menor que 100, receberá nota 1,0;
3. O grupo que apresentar como resultado uma diferença entre a soma dos elementos dos conjuntos menor que 500, receberá nota 0,9;
4. O grupo que apresentar como resultado uma diferença entre a soma dos elementos dos conjuntos menor que 10000, receberá nota 0,8;

5. O grupo que apresentar como resultado uma diferença entre a soma dos elementos dos conjuntos menor ou igual a 32583, receberá nota 0,7;
6. Todos os demais grupos receberão nota 0,6, desde que atendam o item 1 e apresentem resultado menor ou igual a 40000.
7. Grupos com solução maior que 40000 não receberão pontuação no exercício.

No relatório, discuta as etapas de implementação, as estratégias adotadas para se atingir o objetivo e os resultados obtidos.

2. (1,0) Neste exercício o grupo deverá implementar um quantificador de velocidade de digitação do usuário. Tome como exemplo o site: <https://www.typingtest.com/test.html>

Para este exercício é obrigatório o uso de listas encadeadas e alocação dinâmica. Use a menor quantidade de memória possível. Os textos usados para o teste de digitação também estão na pasta da disciplina e são eles: 'digitacao1.txt' e 'digitacao2.txt'. O usuário deve escolher qual deles usar logo no início da execução do programa. O teste deve ter duração de 60 segundos. Não é permitido o uso de bibliotecas gráficas e o programa deve ser executado no console/terminal. Além das bibliotecas citadas no exercício anterior, pode ser utilizada a *time.h*. Qualquer outra biblioteca que precisar ser usada, deverá ter a aprovação do professor.

Ao final do teste, o algoritmo deve mostrar a velocidade de digitação do usuário (palavras por minuto), a quantidade de palavras digitadas erradas e a pontuação final, que é a quantidade de palavras corretamente escritas ao longo de um minuto. Em seguida, mostre ao usuário as palavras do texto original com cores. As palavras digitadas corretamente devem estar em verde enquanto as digitadas de maneira errada, em vermelho. Por conta da limitação do console/terminal, o grupo pode optar por mostrar uma quantidade limitada de palavras, pedindo para o usuário digitar uma tecla específica para ver a segunda parte do texto. Pressionando novamente passa para a próxima parte até que se chegue ao final do DIGITADO PELO USUÁRIO. Observe que se o texto original possuir 15 parágrafos e usuário digitou apenas 2 desses parágrafos, somente esses 2 deverão ser mostrados no resultado.

Apresente no relatório quais foram as estratégias usadas para elaboração do algoritmo bem como onde a(s) lista(s) encadeada(s) e alocação dinâmica foram usadas.

3. (1,0) Ainda no console e usando as bibliotecas apresentadas no primeiro exercício, crie uma agenda eletrônica que mostre os eventos de uma pessoa ao longo de uma semana. Economizar memória é essencial neste programa, portanto, o uso de listas encadeadas e alocação dinâmica é obrigatório. Use cores para diferenciar os dias da semana, bem como os horários em que existem eventos agendados. Implemente um cursor, que o usuário poderá movimentar ao longo da agenda:

- Caso ele posicione o cursor em um horário que não existe um evento agendado, cadastre um novo evento nesse horário e atualize a agenda. Um evento deverá conter obrigatoriamente: uma breve descrição, o horário de início e o local. Por questão de usabilidade, enquanto a agenda é mostrada na tela, apresente apenas a breve descrição no espaço destinado ao horário em que o evento está agendado.
- Caso ele posicione o cursor em um horário já ocupado, peça para o usuário escolher entre excluir o evento ou sobrescrevê-lo com um novo cadastro de evento.

Mantenha os eventos de cada dia ordenados na lista de acordo com o horário de início. Perceba que mais de uma lista pode ser usada. A agenda deverá ter a opção – também acessível pelo cursor – de mostrar todos os eventos de um determinado dia da semana. Quando essa funcionalidade for acionada pelo usuário, peça o dia da semana e mostre todos os eventos agendados para esse dia, incluindo o horário de início, a descrição e local.

Forneça também a opção de gravar a agenda em um arquivo de texto. Neste caso, imprima no arquivo os eventos agendados para cada dia, organizados em ordem cronológica.

Discutir no relatório como foi realizada o processo de cadastro de novos eventos na agenda. Mostrar também os detalhes principais elaborados para a movimentação do cursor e impressão da agenda.