Algoritmos e Estruturas de Dados 1

Segundo Trabalho Prático

Entrega Sugerida: 04/10/2019

2º Semestre 2019 - DC-UFSCar

1 Introdução

Neste trabalho prático TP02 é solicitada a entrega de um programa que solucione o problema apresentado na próxima seção. O arquivo entregue deve seguir os itens abaixo:

- O TP02 deverá ser feito individualmente e plágio não será tolerado;
- O cabeçalho de cada função pedida deve seguir o modelo indicado na sua descrição;
- Você pode adicionar mais bibliotecas caso necessite;
- Outras funções auxiliares podem ser criadas para facilitar o desenvolvimento, desde que as funções pedidas no enunciado estejam presentes e se comportem de acordo com o que foi pedido;
- O TP02 deve ser entregue no run codes (https://run.codes) em um arquivo contendo código em linguagem C e com um cabeçalho com as informações do estudante (nome, curso, RA);
- Cada estudante deve se cadastrar no run codes (https://run.codes) informando Nome Completo, escolhendo "UFSCar - Universidade Federal de São Carlos" no campo Universidade e colocando seu RA no campo Núm. Matrícula. Depois de cadastrado, basta logar no run codes e se matricular na disciplina "1001502 - Algoritmos e Estruturas de Dados 1" usando o Código de Matrícula WAE8.

2 Problema de Josephus

Conta-se uma história sobre o fim de uma guerra hebraica, na qual o historiador Josephus e mais 40 homens foram cercados pelo exército romano. Eles decidiram não se entregar com vida, mas sua religião condena o suicídio. Por isso, fizeram um pacto no qual os 41 homens fariam um círculo e, começando pelo homem na primeira posição, um homem de cada vez mataria aquele imediatamente à sua esquerda, sendo que o próximo homem a matar é o que estava a esquerda do último a morrer. Nesse contexto, Josephus precisou encontrar em qual lugar no círculo deveria se posicionar para que fosse o último homem que restasse.

Essa história dá origem ao **Problema de Josephus** que será o exercício a ser resolvido nesse trabalho.

Para resolver o problema, utilizaremos o conceito de uma lista ligada circular, por isso será necessário implementar algumas funções que auxiliem a criação e manipulação de listas encadeadas, além da resolução do problema em si.

3 Tarefas

Crie um arquivo trabalho2.c contendo um cabeçalho com as informações do estudante, a implementação da lista ligada circular, a resolução do problema de Josephus e um main que simule e resolva o problema.

3.1 Implementação de Lista Ligada

Implemente as funções de **inserção**, **remoção** e **impressão** de elementos em uma lista ligada circular. Os nós da lista devem seguir a struct **pessoa**, a lista deve ser declarada como **inicio** e a assinatura das funções **devem** ser como mostra o seguinte código:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  typedef struct pessoa Pessoa;
5  struct pessoa {
6      int posicao;
7      Pessoa *prox;
8  };
9
10 Pessoa *inicio;
11 inicio = NULL;
12
13 void insereLista (int x);
14 void removeLista (Pessoa *p);
15 void imprimeLista ();
16 int resolveJosephus(int n, int m);
```

É importante seguir o modelo dado acima para possibilitar a correção. Você pode adicionar o que for necessário, como bibliotecas e alterar a ordem de funções, por exemplo, desde que o que existe nesse modelo continue existindo no trabalho enviado.

Trabalharemos com uma **lista encadeada circular** chamada *inicio* que é declarada na linha 10 do modelo, Note que *inicio* é uma variável global, logo você pode acessá-la dentro das funções sem a necessidade de passá-la como argumento. Essa lista conterá elementos que representam a posição dos homens no círculo.

3.1.1 Função: insereLista(int x)

A inserção nessa lista deve acontecer através da função *insereLista(int x)*, em que x é o elemento a ser inserido. O elemento deve ser inserido no **final** da lista e, como trabalharemos com uma lista circular, o último elemento da lista deve sempre apontar para o primeiro elemento dela.

3.1.2 Função: removeLista(Pessoa *p)

Quando um homem morrer, ele deve ser retirado da lista, para isso utilizaremos a função *void removeLista(Pessoa *p)* que deve remover o elemento **p→prox**, ou seja, o elemento imediatamente depois de **p**. Perceba também, que a lista não pode conter elementos iguais uma vez que cada pessoa recebe um número diferente de acordo com sua posição.

3.1.3 Função: imprimeLista()

Essa função não recebe nenhum parâmetro e deve **imprimir todos** os elementos presentes na lista naquele momento, começando por aquele apontado por *inicio*.

3.1.4 Função: main()

A função main deve ler o numero de execuções do problema de Josephus e posteriormente os inteiros n, representando o número de pessoas, e p, representando o tamanho do passo, e devolver o resultado de cada execução. Para isso, deve-se usar a seguinte implementação do programa main.

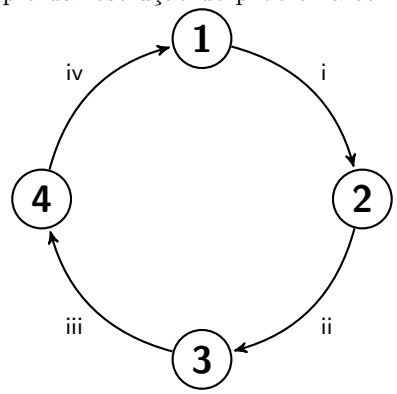
```
int main (){
    int nroexecs;

    scanf("%d", &nroexecs);
    int *n = malloc(nroexecs * sizeof(int));
    int *p = malloc(nroexecs * sizeof(int));

    for(int i = 0; i < nroexecs; i++){
        scanf("%d", &n[i]);
        scanf("%d", &p[i]);
    }

    for(int i = 0;i<nroexecs; i++){
        printf("Usando n=%d, m=%d, resultado=%d\n", n[i], p[
    i], resolveJosephus(n[i], p[i]));
    }

    return 0;
}
```

4 Casos de testes

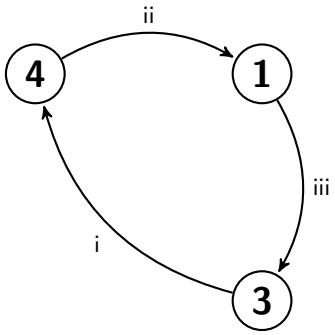
4.1 Solução do Problema de Josephus

Suponha que n pessoas estão dispostas em um círculo e que são numeradas de 1 a n no sentido horário. Comece pela pessoa de número 1 e elimine a m-ésima pessoa. Em seguida, contando a partir da pessoa p que sucede a última eliminada, elimine a pessoa que está m posições depois de p. Repita este processo enquanto o circulo tiver duas ou mais pessoas. Note que, como estamos falando de um círculo, ao andar m posições a partir de uma pessoa, podemos acabar chegando em pessoas que estão antes dela, ou até mesmo a ela própria. Como a pessoa nunca pode matar a si própria, neste caso ela deve matar quem estiver imediatamente à frente dela na lista.

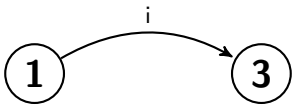
Segue um exemplo da resolução do problema com $n = 4$ e $m = 1$:



A primeira pessoa a ser eliminada é a que se encontra na posição 2.



A próxima pessoa eliminada se encontra na posição 4.



A última pessoa eliminada se encontra na posição 3 e a sobrevivente é que se encontra na posição 1.

Sua função deve receber como parâmetros os inteiros n, que representa o número de pessoas no círculo, e m, que representa o tamanho do passo (número de pessoas puladas no círculo) para a remoção do próximo. Ela deve **imprimir** o número do sobrevivente.

4.2 Exemplo de Caso de Teste

Entrada:

```
3
10
1
10
2
10
3
```

Saída:

```
Usando n=10, m=1, resultado=5
Usando n=10, m=2, resultado=10
Usando n=10, m=3, resultado=6
```

5 Referências interessantes

Josephus Problem - Wikipedia: `https://en.wikipedia.org/wiki/Josephus_problem`

The Josephus Problem - Numberphile (vídeo) `https://goo.gl/MVNcB6`

Site Projeto de Algoritmos: `https://www.ime.usp.br/~pf/algoritmos/`