Universidade Federal do Ceará
Departamento de Computação
Fundamentos de Programação
1° Avaliação Parcial

Professor Wladimir Araújo Tavares

Parte I - Inspeção de Listas

1. Soma de Sequência de ímpares 1.0

   Imagine que nós temos uma lista de números armazenada na memória

	0	1	···							N-1
L	17	10	6	15	9	3	42	20	16	25	5

   A tarefa consiste em

   → *Encontrar a maior sequência de ímpares consecutivos,*
   e armazenar o seu tamanho na variável `resp`

   No exemplo acima, isso nos daria

   ```
   =>  resp = 3
   ```

   Apresente um programa que realiza essa tarefa.

   Nota: Caso a lista não contenha números ímpares, `resp` deve receber o valor 0.

2. Elementos incomuns

   Imagine que nós temos duas listas ordenadas U e V.

   Por exemplo,

	0	1	···								N-1
U	1	3	4	8	11	12	13	18	26	28	30	33

	0	1	···								N-1
V	3	4	6	8	13	16	18	26	27	30	41	42

   A tarefa consiste em contar a quantidade de elementos que aparecem em apenas uma das listas

   No exemplo acima, isso nos daria

   =¿ 9 elementos: 1, 6, 11, 16, 27, 28, 33, 41, 42

   Apresente um programa que realiza essa tarefa.

   Dica: Percorra a lista U da esquerda para a direita. E vá deslizando um dedo em V pulando os elementos menores que o elemento apontado por U.

```c

#define N 10
int U[N] = {1,3,4,8,11,12,13,18,26,28,30,33};
int V[N] = {3,4,6,8,13,16,18,26,27,30,41,42};
int cont;

int main(){
    // Faça seu programa aqui


}
```

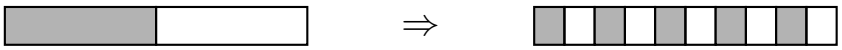
Parte II - Inspeção da lista

Intercalação

Imagine que nós temos uma lista de tamanho `N` (par).

A tarefa consiste em

→ *Colocar os elementos da primeira metade nas posições pares da lista*
e colocar os elementos da segunda metade nas posições ímpares da lista



Por exemplo, se a nossa lista é

L	1	2	3	4	5	6	7	8	9	10

então isso nos daria

L	1	6	2	7	3	8	4	9	5	10

Faça um programa que realiza essa tarefa.

Dica: Você pode utilizar uma lista auxiliar se quiser.

```c
#define N 10
int L[N] = {8,2,5,11,29,13,6,16,7,10};
int a = 5, b = 15;
int cont;
int main(){
    // Escreva seu código
}
```

3. (**Jogo do Sapo**) Imagine que nós temos uma lista de números inteiros L representando a altura canos e um inteiro P representando a altura do pulo do sapo. Você quer saber se o sapo começando em cima do cano mais à esquerda consegue chegar a salvo no cano mais à direita. Observe que o sapo só consegue sobreviver se a diferença de altura entre canos consecutivos for no máximo a altura do pulo do sapo. Caso contrário, a sapo bate no cano e morre.

   Sua tarefa é dado uma lista L representando as alturas dos canos e um inteiro P, representando a altura do sapo, verificar se o sapo consegue chegar em segurança no cano mais à direita. se isso for o caso, a variável resp deve receber o valor 1, caso contrário ela deve receber o valor 0.

   Por exemplo, dado a lista L

	0	1	2	3	4	5	6	7	8	9
L	1	3	6	9	7	2	4	5	8	3

   Se $P = 5$, o sapo consegue chegar em segurança no cano mais à direita. Se $P = 4$, o sapo não consegue fazer o salto ($7 \rightarrow 2$).

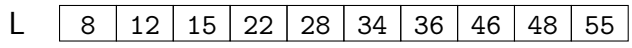
Salto	Diferença de Altura
1 (1 → 3)	2
2 (3 → 6)	3
3 (6 → 9)	3
4 (9 → 7)	2
5 (7 → 2)	5
6 (2 → 4)	2
7 (4 → 5)	1
8 (5 → 8)	3
9 (8 → 3)	5

```c
#define N 10
int L[N] = {1,3,6,9,7,2,4,5,8,3};
int P = 5;
int resp;
int main(){
    //Escreva seu código
}
```

Parte III - Manipulação de Listas

4. Dividindo o maior par

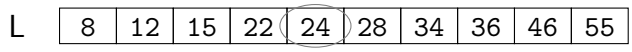
Imagine que nós temos uma lista ordenada armazenada na memória.



A tarefa consiste em

→ *Localizar o maior elemento par, dividi-lo por 2,*
e move-lo para a posição correta na lista ordenada

No exemplo acima, isso nos daria

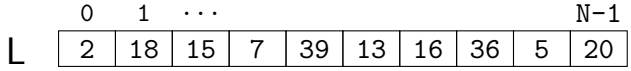


Apresente um programa que realiza essa tarefa.

```c
#define N 10
int L[N] = {8,12,15,22,28,34,36,46,48,55};
int main(){
    //Escreva seu código aqui
}
```

5. Troca-troca

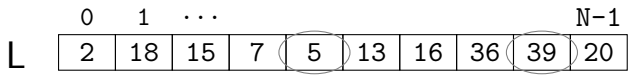
Imagine que nós temos uma lista de números armazenada na memória.
Por exemplo,



Agora imagine que a tarefa é a seguinte

→ *Localizar o menor elemento ímpar da lista,*
localizar o maior elemento ímpar da lista,
e trocar esses dois elementos de lugar (um com o outro)
caso não exista dois ímpares na lista a variável trocou recebe 0 caso contrário recebe 1.

No exemplo acima, isso nos daria



Apresente um programa que realiza essa tarefa.

Nota: Você pode assumir que todos os elementos da lista são distintos.

Nota: Caso não existam dois elementos ímpares na lista, o programa deve indicar esse fato.

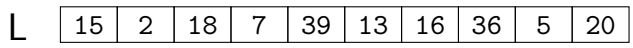
```c
#define N 10
int L[N] = {2,18,15,7,39,13,16,36,5,20};
int resp;
int main(){
    //Escreva seu código aqui
}
```

Parte IV - Desafio

6. Empurrando os ímpares

Imagine que nós temos uma lista de números armazenada na memória, onde o último elemento é par.

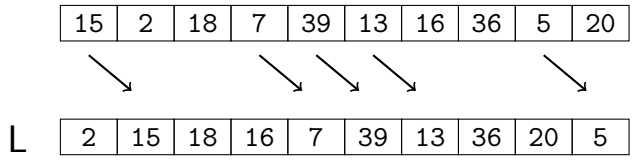
Por exemplo,



A tarefa consiste em

→ *Empurrar todos os elementos ímpares 1 posição para a direita*
mantendo os demais elementos na mesma ordem

No exemplo acima, isso nos daria




```c
#define N 10
int L[N] = {15,2,18,7,39,13,16,36,5,20};
int resp;
int main(){
    //Escreva seu código aqui
}
```