UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA

Disciplina: [1107186] Estrutura de Dados
Prof. Christian Azambuja Pagot.
Data:
Nome: _____ Matrícula: _____

Prova I

Instruções:

- Esta prova tem duração de 2 horas.
- A prova é individual e sem consulta.
- Antes de iniciar a prova, escreva seu nome e número de matrícula nas folhas da prova e de respostas.
- A interpretação dos enunciados faz parte da verificação.
- A prova pode ser feita a lápis, porém as respostas devem estar a caneta.
- É vedado o uso de equipamentos eletrônicos (celulares, calculadoras, etc.) durante o período da prova.
- As questões que envolvem cálculos devem apresentar todo o desenvolvimento dos mesmos.
- Questões que envolvem a escrita de funções e de programas devem apresentar o código completo (nome de funções, parâmetros, tipos, valores de retorno, declaração de variáveis, etc.).

Questão 1 (2,5 pt)

Vimos em aula que uma fila pode ser implementada com o auxílio de um *array*. Um exemplo de implementação, em C++, de uma fila de até 5 números inteiros positivos, utilizando *array*, é apresentada a seguir:

```
class Queue {
private:
       int queue[5];
       int first;
       int last;
public:
      ...
       void push(int val);
       int pop(void);
};

Queue::Queue() {
       this->first = 0;
       this->last = -1;
};
```

```
void Queue::push(int val) {
       this->last++;
       this->queue[this->last] = val;
}

int Queue::pop(void) {
       int val;
       if (this->last >= this->first)
       {
              val = this->queue[this->first];
              this->first++;

              return val;
       }
       else
              return -1;
}
```

Como se pode observar, a capacidade de armazenamento desta fila é limitada devido à utilização de um *array* de tamanho fixo. Extenda a implementação acima de forma a suportar o armazenamento de filas de tamanhos arbitrários. **Importante:** A extensão deve continuar armazenando os dados da fila em *arrays*.

Questão 2 (2,0 pt)

Como visto em aula, uma boa função de *hash* deve distribuir as chaves de maneira uniforme entre os *buckets* da tabela hash. Com base nisto, responda as perguntas a seguir:

1. Considere uma tabela *hash* contendo *n buckets*, e que deseja-se inserir *m* elementos nesta tabela. Responda: Qual o número de colisões por *bucket* que uma função de *hash* deve, idealmente, gerar neste caso? Explique sua resposta.
2. Considere uma tabela de *hash* contendo 13 buckets (indexados de 0 à 12), e uma função de *hash* associada, tal que *hash* = *chave* mod 13. Esta função de *hash* pode, de acordo com o que foi visto em aula, ser considerada uma boa função de *hash*? Explique sua resposta e a ilustre com exemplos.

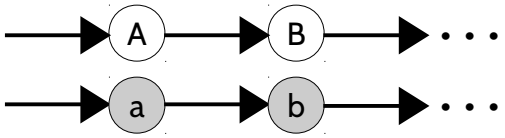
Questão 3 (3,0 pt)

Escreva uma função recursiva, em C/C++, que verifique se uma palavra é palíndrome. Abaixo segue o protótipo da função a ser desenvolvida, onde ***word*** é um *array* de *char* contendo a palavra, e ***size*** é o número de chars (desconsiderando-se o '\0'). Se a palavra for palíndrome, a função deve retornar *true*, caso contrário deve retornar *false*.

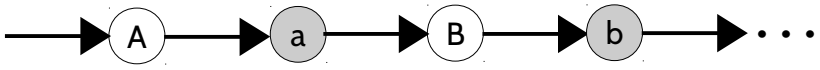
```
bool isPalindrome(char* word, int size)
```

Questão 4 (2,5 pt)

Considere duas listas simplesmente encadeadas, uma com *n* elementos e a outra com *m* elementos. Escreva uma função em C/C++ que faça a combinação (*merge*) das duas listas, combinando os nós das listas originais de forma que eles se alternem na lista resultante. **Importante:** A função não deve criar novos nós! Exemplo: Supondo as seguintes listas a serem combinadas



Após o *merge*, a lista resultante deve ficar:



Assuma que a estrutura de cada nó é dada pelo seguinte trecho de código:

```
struct Node {
       char value;
       struct Node next;
}
```

Abaixo segue o protótipo da função a ser desenvolvida:

```
void Merge(struct Node* l1, struct Node* l2)
```