1ª Avaliação Parcial
Curso: Engenharia da Computação
Disciplina: Estruturas de Dados
Prof. Jarbas Joaci de Mesquita Sá Junior
Universidade Federal do Ceará – UFC/Sobral

Nome: Francisco Ruan Gomes Damasceno Data 02 / 05/2022

1ª) Qual a finalidade da função abaixo e qual a sua complexidade computacional no pior e no melhor caso? (1,0 ponto)

```
void xxxx(int v[], int n){
  int i,j;
  for (i=n-1;i>=1;i--)
    for (j=0;j<i;j++){
      if (v[j]>v[j+1]){
        int temp = v[j];
        v[j]=v[j+1];
        v[j+1]=temp;
      }
    }
}
```

0,7

2ª) Considere que um nó de uma lista encadeada é dado por:

```
typedef struct lista Lista;
struct lista {
  int info;
  Lista *prox;
};
```

a) Implemente uma função que tenha como valor de retorno a quantidade de nós de uma lista encadeada que possuem valores maiores que x e menores que y. O protótipo da função deve ser: (1,0 ponto)

```
int qtd_lista(Lista* l, int x, int y);
```

0,9

b) Implemente uma função que insere um valor num **novo nó** no **fim** de uma lista. O protótipo da função deve ser: (2,0 pontos)

```
Lista* insere_fim_lista(Lista* l, int x);
```

1,0

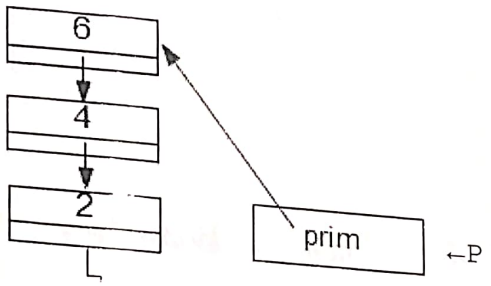
3ª) Considere uma pilha implementada por meio de listas encadeadas, conforme as descrições de estruturas abaixo:

```
typedef struct lista Lista;

typedef struct pilha Pilha;

struct lista{
   int info;
   Lista *prox;
};

struct pilha{
   Lista *prim;
};
```



Exemplo de Pilha

a) Escreva uma função que acrescente um elemento ao topo da pilha apenas se ele for **maior** que o topo já existente. O protótipo da função deve ser: (2,0 pontos)

```
void pilha_push_maior(Pilha *p, int info);
```

Obs. se a pilha estiver inicialmente vazia, qualquer valor será aceito como topo da pilha.

b) Escreva uma função que retire um elemento do topo da pilha e retorne o seu valor **apenas se ele for par (caso contrário, a pilha permanece inalterada).** O protótipo da função deve ser: (2,0 pontos)

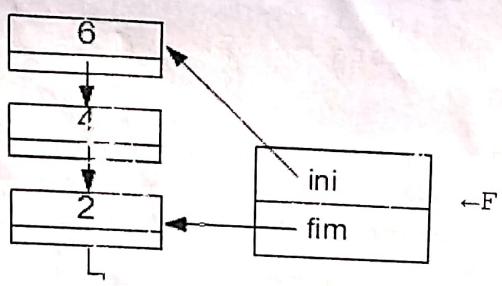
```
int pilha_pop_par(Pilha *p);
```

4ª) Considere uma fila implementada por meio de listas encadeadas, conforme as descrições de estruturas abaixo:

```
typedef struct lista Lista;
typedef struct fila Fila;

struct lista{
   int info;
   Lista *prox;
};

struct fila{
   Lista *ini;
   Lista *fim;
};
```



Exemplo de Fila

Escreva uma função que retire um elemento do início da fila e que retorne o seu valor **apenas se ele for par (caso contrário, a fila permanece inalterada).** O protótipo da função deve ser: (2,0 pontos)

```
int fila_retira_par(Fila *f);
```

02/05/22

Nom: Francisco Ruan Gomes Damasceno

1- A função tem como funcionalidade a ordenação do vetor. Analisando e observando o algoritmo, obtemos que a complexidade de um "for" é dada por O(n). como temos dois observamos que sua complexidade é $O(n^2)$, pois assumimos que se um "for" for executado n vezes o segundo segue a mesma ordem pois estão diretamente ligados. Assumindo assim o pior caso. Considerando que ambos "for" estão ligados e considerando um caso favorável onde o vetor já esteja ordenado vemos obter $\Omega(n)$ como melhor caso.

2:

```
a) int qtd_lista(lista* l, int x, int y){
    if(l==NULL){
      Printf("Lista Vazia\n");
        exit(1);
    } int cont=0;
    lista* ln=l;
    while(ln!=NULL){
      if((ln->info>x) && (ln->info<y))
          cont++;
      ln=ln->prox;
    } return cont;
    }
```

4/09

Restante do 2:

b)

```
Lista* insere_fim_lista(Lista* l, int x) {
  if(l==NULL) {
      Printf("Lista Vazia\n");
      exit(3);
  }
  Lista* ln = (Lista*)malloc(sizeof(Lista));
  if(ln == NULL){
     Printf("Memória insuficiente\n");
      exit(3);
  } Lista* laux=l;
  while(laux!=NULL) {
      laux = laux->prox;
    }
  ln->info = x
  ln->prox = NULL;
  laux->prox = ln;
}
```

Função deve insere elemento na lista vazia

laux->prox

3.º

a)

```
Void pilha_push_maior(Pilha *P, int info){
  if((P->prim->info)>info || P->prim == NULL){
    Lista* ln = (Lista*)malloc(sizeof(Lista));
    if(ln==NULL){
        Printf("Memória insuficiente"); exit(3);
    } ln->info=info;
    ln->prox=P->prim;
    P->prim=ln;
  } else {
        Printf("O termo não é maior\n");
  }
}
```

b)
```
int Pilha_pop_par(Pilha *p){
    if(p->prim==NULL){
        Printf("Pilha Vazio");
        exit(1);
    } else if((p->prim->info)%2==0){
        Lista *ln = p->prim;
        int a = p->info;
        p->prim = p->prim->prox;
        free(ln);
    return a; }}
```

2:
```
int fila_retira_par(Fila *f){
    if(f->ini==NULL){
        Printf("Pilha Vazia");
        exit(1);
    } else if((f->ini->info)%2==0){
        Lista *ln = f->ini;
        int a = ln->info;
        f->ini = f->ini->prox;
    free(ln);
        if(f->ini==NULL)
            f->fim=NULL;
    return a;
} else {
        Printf("O turno é difunto de para/n");
    }
}
```

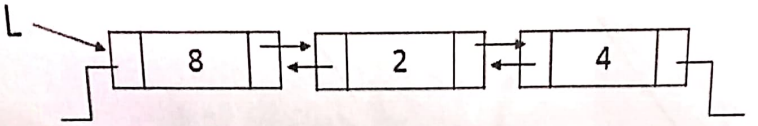
2ª Avaliação Parcial
Curso: Engenharia da Computação
Disciplina: Estruturas de Dados
Prof. Jarbas Joaci de Mesquita Sá Junior
Universidade Federal do Ceará – UFC/Sobral

Nome: Francisco Ruan Gomes Damasceno Data 06/06/2022

1ª) Considere que um nó de uma lista duplamente encadeada é dado por:

```
typedef struct lista_dupl ListaDupl;
struct lista {
  int info;
  ListaDupl *ant;
  ListaDupl *prox;
};
```

Ex:



Implemente uma função que insere um valor em um **novo nó** no **fim** de uma lista duplamente encadeada. Essa função deverá receber o endereço do começo da lista e o valor a ser inserido, e deverá retornar o endereço do primeiro nó da lista. O protótipo da função deve ser: (2,0 pontos)

```
Lista* insere_fim_lista_dupl(ListaDupl* l, int x);
```

2ª) Realize os seguintes procedimentos:

a) Insira a seguinte sequência de chaves em uma árvore binária de busca: 39, 22, 13, 65, 44, 32, 56, 41, 17. (1,0 ponto);

b) Remova os elementos 22, 17 e 13 (1,0 ponto). **Obs**: a remoção deve obrigatoriamente ocorrer na ordem apresentada.

3ª) Insira a seguinte sequência de chaves em uma árvore AVL: 19, 22, 13, 55, 44, 12, 56, 11, 17. (2,0 pontos)

4ª) Construa uma árvore rubro-negra para a seguinte sequência de chaves: 31, 62, 33, 42, 57, 68, 17, 81, 93. (2,0 pontos)

5ª) Explique como funciona o algoritmo **quicksort**. Quais são suas complexidades no pior e no melhor caso? (2,0 pontos)

06/06/22

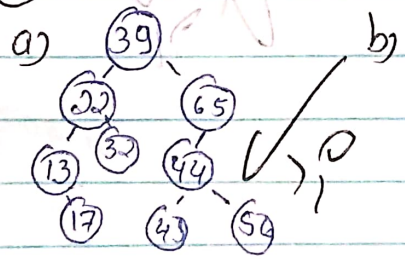
Nome: Francisco Ruan Gomes Damasceno

1)

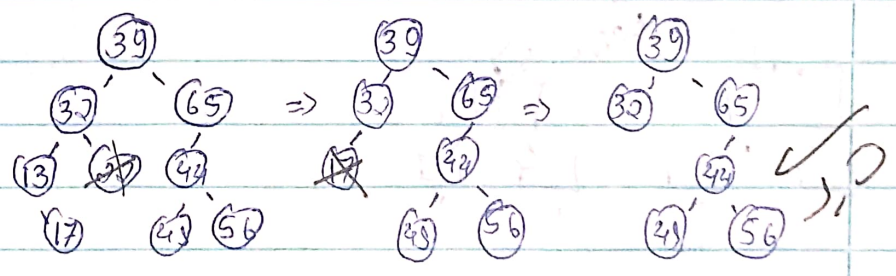
```
ListaDupl* inserir_fim_lista_dupl(ListaDupl* l, int x){
    ListaDupl* ln = (ListaDupl*) malloc(sizeof(ListaDupl));
    ln->info = x;
    ln->prox = NULL;
    ln->ant = l;
    if(l == NULL)
        return ln;
    ListaDupl* laux = l;
    while(laux->prox != NULL){
        laux = laux->prox;
    }
    laux->prox = ln;
    ln->ant = laux;
    return l;
}
```

2)

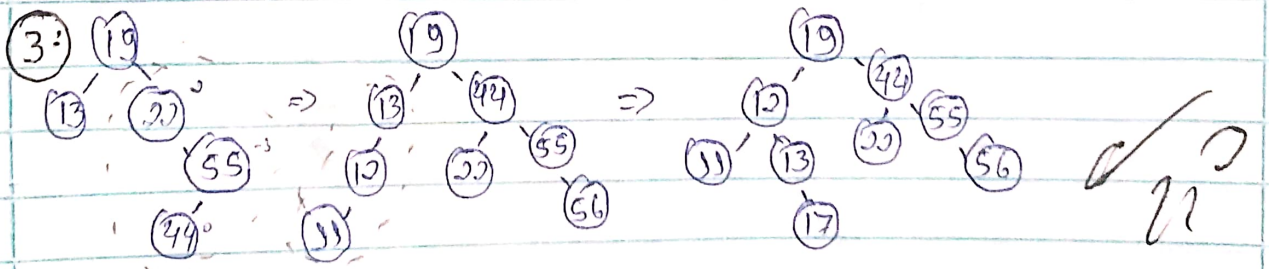
a)



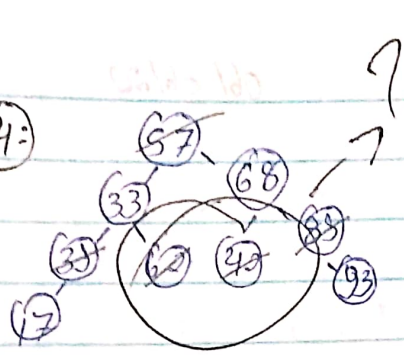
b)



3)



4:



5: O algoritmo funciona de modo que através de uma função, declaramos variáveis do tipo inteiro a, b para percorrer todo o vetor que já foi instanciado. Através de um elemento desse vetor atribuimos o seu valor como um pivô, as variáveis a, b percorrem o vetor ordenando ele de modo que deixa o mesmo em uma posição no qual obtemos dois sub-vetores um com os elementos a direita do pivô como maiores e a esquerda como menores esses sub-vetores são chamados na função de forma recursiva duas vezes uma para o sub-vetor direita e outro esquerda e com esses valores percorremos todo o vetor executando todos os passos, ao fim retornamos cada valor de modo a ficar ordenado, essa recursão é feita de modo que reste 0 ou 1 elementos no vetor. Sua complexidade é dada por $O(n^2)$ para o pior caso e $O(n \log n)$ para o melhor caso e caso médio