unesp

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de Bauru

Faculdade de Ciências

1ª. Prova (P1)

24/Abril/2019

Questão 01		2,5
Questão 02		2,5
Questão 03		1,0
Questão 04		1,0
Questão 05		3,0
Total		**10,0**

Disciplina: Estruturas de Dados I – **ED1**
Professora: Dra Simone das Graças Domingues Prado

Nome:_____ **RA:** _____

(Questão 01) (URI Online Judge | 1522-modificado) Claudio inventou um novo jogo, chamado de *Jogo das pilhas*, e quer submetê-lo ao próximo concurso de jogos da URI (União Recreativa Internacional). Apesar de muito divertido, o jogo parece ser muito difícil de ganhar, logo Claudio pediu sua ajuda para avaliar se algumas instâncias do jogo podem ser vencidas. O *jogo das pilhas* é individual, e é jogado com três pilhas, inicialmente com o mesmo número de cartas. Cada carta tem um valor numérico inteiro de 0 até 9. O jogador pode, a qualquer momento ver o valor de qualquer carta, mas só pode jogar com as cartas que estão no topo das pilhas. Em cada rodada, o jogador pode remover qualquer combinação de cartas que estejam no topo da pilha (pode escolher 1, 2 ou até 3 cartas) cuja soma dos valores seja múltipla de 3. **O jogo é ganho quando todas as cartas forem removidas das pilhas. Se alguma carta não puder ser removida, perde-se o jogo**.

(a) (0,2pt) Use o tipo de pilha que preferir (estática ou dinâmica). Somente a declare.
(b) (0,3pt) Suponha que as rotinas de empilha e desempilha já estejam prontas. Escreva somente o cabeçalho delas.

Entrada

A entrada é composta por várias instâncias. Cada instância é iniciada por um inteiro **N** ($0 \leq \mathbf{N} \leq 100$), que identifica o número de cartas em cada pilha. A entrada termina quando **N** = 0. Cada uma das **N** linhas seguintes contém três inteiros **A**, **B** e **C**, que descrevem os valores numéricos das cartas em um nível da pilha ($0 \leq \mathbf{A}, \mathbf{B}, \mathbf{C} \leq 9$). As pilhas são descritas do topo até o fundo.

Saída Modificada.

(c) (2,0pt) Faça uma rotina para que ao receber os dados de entrada (N e as três pilhas), retorne 1 (se o jogador pode ganhar) ou zero (caso contrário). Considere que sempre será retirada uma carta de cada uma das três pilhas de cartas. Se não usar as três cartas, devolva-as às pilhas correspondentes.

```
int jogo (int N, def_pilha *P1, def_pilha *P2, def_pilha *P3){
      int a,b,c,ganhando=0, parar=0, desempilhou1, desempilhou2,desempilhou3;

      while (!parar){ // termina o jogo se não puder fazer combinação das cartas retiradas
            desempilhou1=desempilhou2=desempilhou3=1;
            if(!desempilha(P1,&a)) desempilhou1=0;
            if(!desempilha(P2,&b)) desempilhou2=0;
            if(!desempilha(P3,&c)) desempilhou3=0;
            if ((desempilhou1&&desempilhou2&&desempilhou3) && ((a+b+c)%3 == 0))
                  ganhando+=1;
            else if ((desempilhou1 && desempilhou2) && ((a+b)%3==0)){
                        ganhando+=1; if(desempilhou3)empilha(P3,c);}
            else if ((desempilhou1&&desempilhou3)  && ((a+c)%3==0)){
                        ganhando+=1; if(desempilhou2)empilha(P2,b);}
            else if ((desempilhou2&&desempilhou3) && ((b+c)%3==0)){
                   ganhando+=1; if(desempilhou1)empilha(P1,a);}
```

Departamento de Computação
Av. Engº Luiz Edmundo Carrijo Coube 14-01 Vargem Limpa Bauru-SP CEP: 17033-360
Fone: (14) 3103-6034 Fax: (14) 3103-6079 email: dcogeral@fc.unesp.br

```
            else if(desempilhou1 &&  a%3 ==0){
                        ganhando+=1; if(desempilhou2)empilha(P2,b);
                        if(desempilhou3)empilha(P3,c);}
            else if(desempilhou2 && b%3 ==0){
                        ganhando+=1;if(desempilhou1)empilha(P1,a);
                        if(desempilhou3)empilha(P3,c);}
            else if(desempilhou3 && c%3 ==0){
                        ganhando+=1; if(desempilhou1)empilha(P1,a);
                        if(desempilhou2)empilha(P2,b);}
            else{ parar=1;
                  if(desempilhou1)empilha(P1,a);
                  if(desempilhou2)empilha(P2,b);
                  if(desempilhou3)empilha(P3,c);}
            if(vazia(*P1)&&vazia(*P2)&&vazia(*P3)) parar=1;
            }
      if ((ganhando==N) && vazia(*P1) && vazia(*P2) && vazia(*P3)) return 1;
      return 0;
}
```

(Questão 02) (URI Online Judge | 2065-modificado) Hoje é a inauguração de um grande supermercado em sua cidade, e todos estão muito excitados com os baixos preços prometidos. Este supermercado quer implantar um método de atendimento para otimizar o tempo do cliente. O esquema é o seguinte:
- inicialmente todos os clientes entram numa fila única.
- depois são separados em duas outras filas: uma para pessoas com poucas mercadorias (menos de 10 itens) ou muitas mercadorias.

Há 02 caixas abertos o tempo todo. Há **M** clientes na fila para serem atendidos, cada um com um determinado número de itens na sua cesta. Dadas essas informações, o gerente pediu sua ajuda para descobrir quanto tempo levará para que todos os clientes sejam atendidos. Ele quer também saber quanto tempo levou cada fila individualmente.
(a) (0,2pt) Use o tipo de fila compatível (estática ou dinâmica desde que seja ao contrário ao que foi usado no exercício 01, ou seja, se fez uma pilha estática, use uma fila dinâmica. Se usou uma pilha dinâmica, faça com fila estática). Somente a declare.
(b) (0,3pt) Suponha que as rotinas de enfileira e desenfileira já estejam prontas. Escreva somente o cabeçalho delas.

Entrada

A primeira linha conterá **M**, indicando o número de clientes, e **T**, tempo médio de processamento de um item comprado. Em seguida haverá **M** inteiros $\mathbf{c_j}$, indicando quantos itens o **j**-ésimo cliente tem em sua cesta ($1 \leq \mathbf{c_j} \leq 100$, para todo $1 \leq \mathbf{j} \leq \mathbf{M}$).

Saída

(c) (2,0pt) Faça uma rotina que receba os dados acima (M, T e $\mathbf{c_j}$), crie e mostre as duas filas separadas (pela qtdade de itens). Devolva quanto tempo levará para que todos os clientes sejam atendidos, bem como quanto tempo levou para cada uma das 2 filas serem totalmente atendida.

Resolução:

```
      void caixas (int tempo, def_fila *F){
            int item,cont,cont1=0,cont2=0;
            def_fila Fmuita,Fpouca;
            Fmuita = Fpouca=NULL;
```

```
        while (desenfileira(F,&item)!=0){
                if (item>10) enfileira (&Fmuita,item);
                else enfileira(&Fpouca, item);}
        printf("\nFila com muita mercadoria: ");imprime(Fmuita);
        printf("\nFila com pouca mercadoria: ");imprime(Fpouca);
        while (desenfileira(&Fmuita,&item)!=0) cont1 += item*tempo;
        while (desenfileira(&Fpouca,&item)!=0) cont2 += item*tempo;
        cont = cont1 + cont2;
        printf("\n A fila com muita mercadoria demorou %d segundos.",cont1);
        printf("\n A fila com pouca mercadoria demorou %d segundos.",cont2);
        printf("\n A fila geral demorou %d segundos.",cont);
 }
```

(Questão 03) Usando a definição da árvore genérica de Grau 3 abaixo, construa as rotinas solicitadas:

```
typedef struct no{
    int info;
    struct no* filho[3];
    } *def_arvore;
```

a) **(0,5pt)** que receba uma árvore genérica e devolva a quantidade de nós que possuem todos os filhos.

```
int conta_nos_todos_filhos(def_arvore arvore){
      int i,cont=0;

      if(arvore==NULL) return 0;
      for (i=0; i<Grau;i++)
            if(arvore->filhos[i]!=NULL)
                  cont+= conta_nos_todos_filhos(arvore->filhos[i]);
      if (arvore->filhos[0]!=NULL && arvore->filhos[1]!=NULL &&
          arvore->filhos[2]!=NULL) return (cont+1);
   return(cont);
}
```

b) **(0,5pt)** que receba uma árvore genérica e devolva a quantidade de nós que são maiores que a raiz.

```
int conta_nos_maiores(def_arvore arvore, int valor){
  int i,cont=0;
  if(arvore==NULL) return 0;
  for(i=0;i<Grau;i++)
      if(arvore->filhos[i]!=NULL)cont+=conta_nos_maiores(arvore->filhos[i],valor);
  if (arvore->info > valor)  return(cont + 1);
  return(cont);
}
```

(Questão 04) (1,0pt) Considere uma árvore binária. Calcule a soma dos elementos que possuem dois filhos. Por exemplo, o resultado para essa árvore seria: 50 + 30 + 70 + 80 = 230

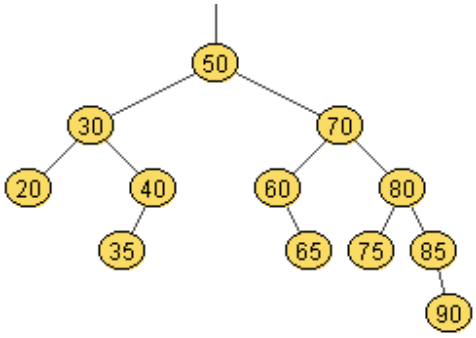


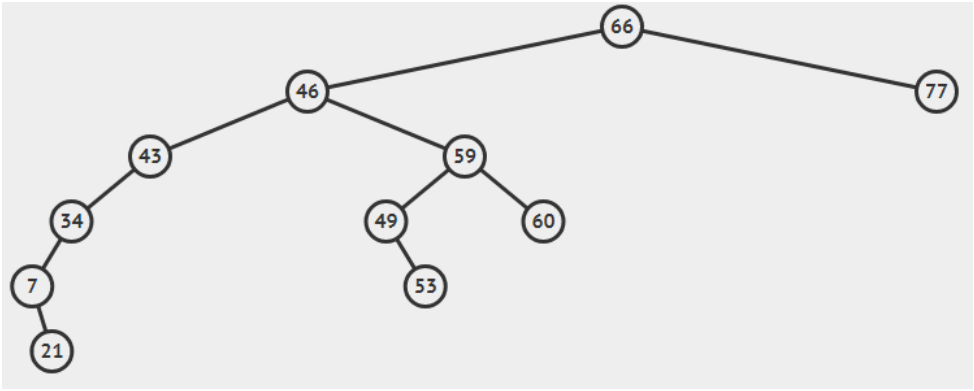
Figura 1. Uma árvore Binária

```
int soma_dois_filhos(def_arvore arvore){
   if(arvore==NULL) return 0;
   if(arvore->dir!=NULL && arvore->esq!=NULL)
      return(arvore->info +
             soma_dois_filhos(arvore->dir) + soma_dois_filhos(arvore->esq));
   return (soma_dois_filhos(arvore->dir)+soma_dois_filhos(arvore->esq));
}
```

(Questão 05) Considere o uso de uma árvore binária de busca

(a) **(1,1pt)** Construa, de forma gráfica, a árvore binária de busca a partir da sequencia

S = 66, 46, 59, 60, 49, 77, 43, 34, 53, 7, 21;



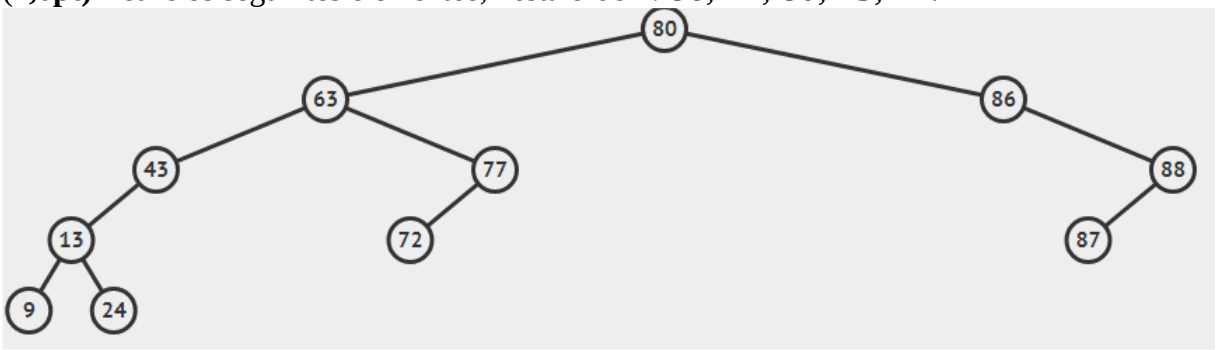
(b) **(0,9pt)** Mostre as sequencias obtidas pelos percursos

PRÉ-ORDEM (rED): 66, 46, 43, 34, 7, 21, 59, 49, 53, 60, 77

EM-ORDEM (ErD): 7, 21, 34, 43, 46, 49, 53, 59, 60, 77

PÓS-ORDEM (EDr): 21, 7, 34, 43, 53, 49, 60, 59, 46, 77, 66

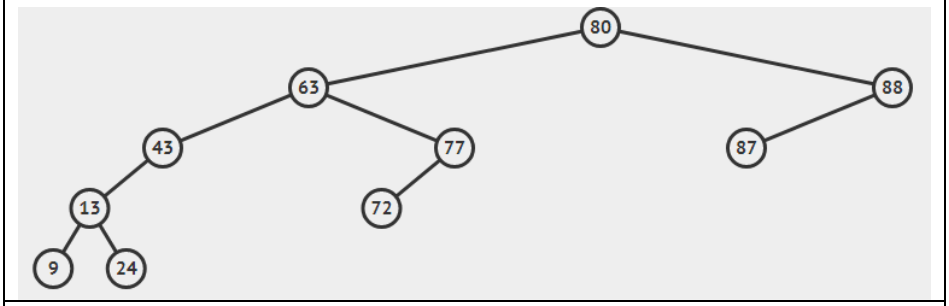
(c) (**1,0pt**) Retire os seguintes elementos, nesta ordem: 86, 77, 80, 43, 72.



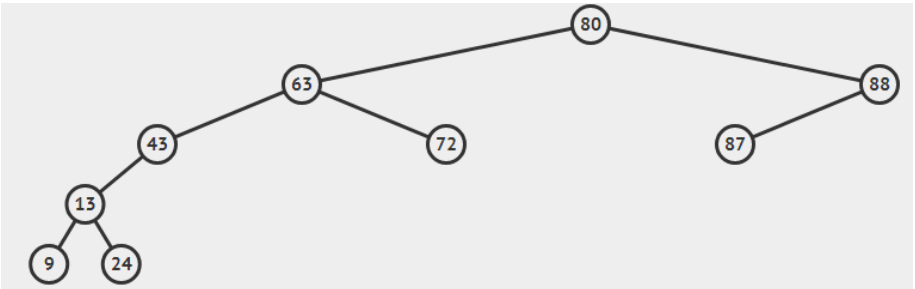
Departamento de Computação
Av. Engº Luiz Edmundo Carrijo Coube 14-01 Vargem Limpa Bauru-SP CEP: 17033-360
Fone: (14) 3103-6034 Fax: (14) 3103-6079 email: dcogeral@fc.unesp.br

1ª Opção

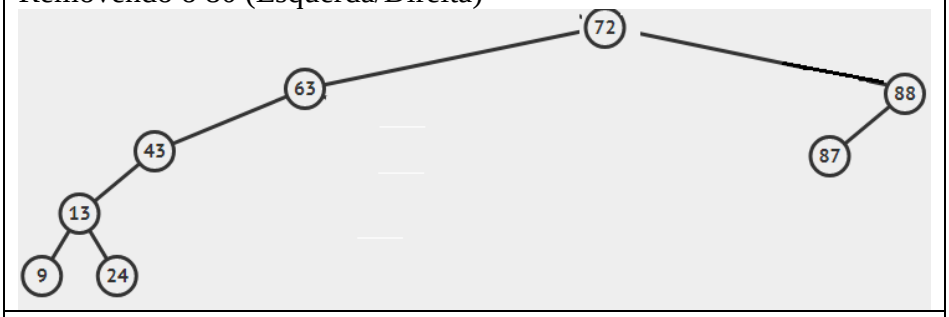
Removendo o 86



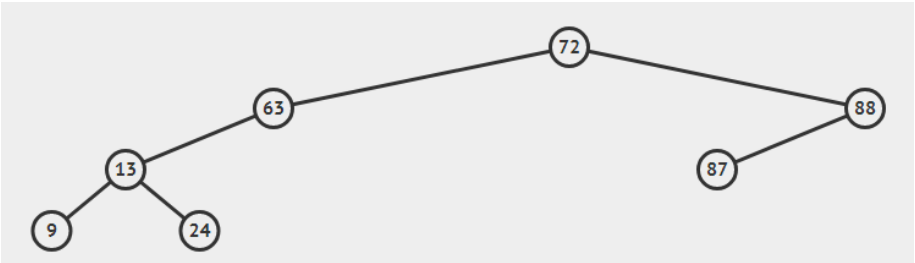
Removendo o 77



Removendo o 80 (Esquerda/Direita)



Removendo o 43

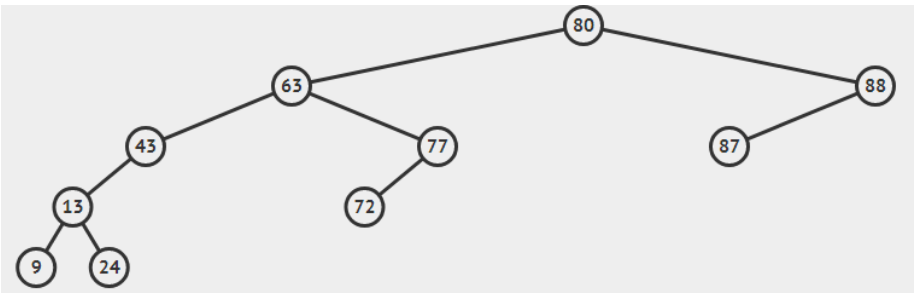


Removendo o 72 (Esquerda/Direita)

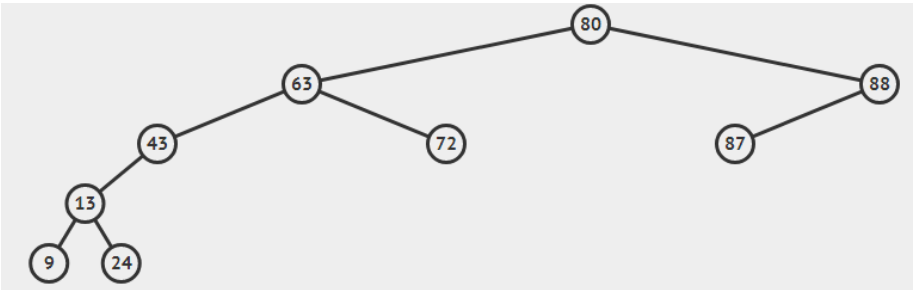
Departamento de Computação
Av. Engº Luiz Edmundo Carrijo Coube 14-01 Vargem Limpa Bauru-SP CEP: 17033-360
Fone: (14) 3103-6034 Fax: (14) 3103-6079 email: dcogeral@fc.unesp.br

2ª Opção

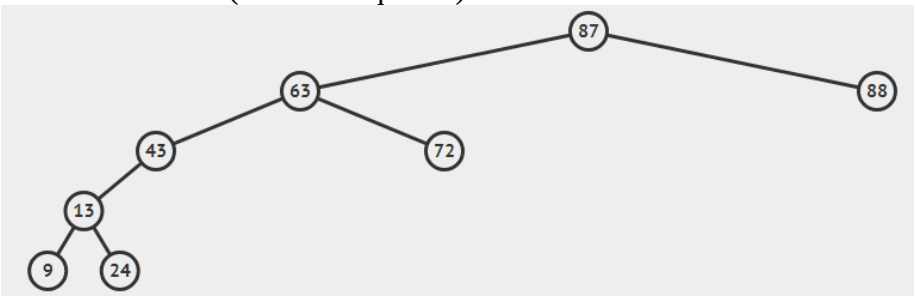
Removendo o 86



Removendo o 77

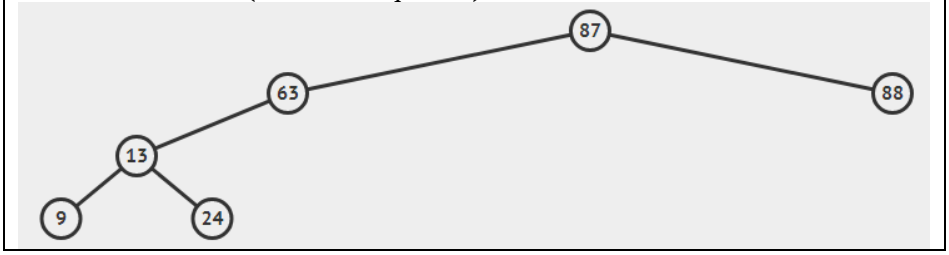


Removendo o 80 (Direita/Esquerda)



Removendo o 43

Removendo o 72 (Direita/Esquerda)



Departamento de Computação
Av. Engº Luiz Edmundo Carrijo Coube 14-01 Vargem Limpa Bauru-SP CEP: 17033-360
Fone: (14) 3103-6034 Fax: (14) 3103-6079 email: dcogeral@fc.unesp.br