INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA - IFMA
Curso: Sistemas de Informação
Algoritmos e Estrutura de Dados – Prof. Gentil Cutrim Serra Jr
2ª Avaliação – 16/Novembro/2022

Aluno: _____

1) [2pt] Considere uma **<u>função de implementação</u>** capaz de remover a segunda metade dos elementos de uma fila. Sem utilizar as operações de TAD, escreva essa função para uma fila implementada em alocação dinâmica de memória. Lembre-se de utilizar a função **`free`** para liberar espaços de memória.

```
int remove(tipo_fila *f) {
   ptr_nodo aux, del;
   int tam = tamanho(*f);
   aux = f->inicio;
   for (int i=1; i<tam; i++)
      aux = aux->prox; // aponta para o último da 1a metade
   del = aux->prox;  // aponta para o primeiro da 2a metade
    aux->prox = NULL; // o último da 1a metade torna-se o último
da fila
   aux = del;
   while (aux != NULL) {
      aux = aux->prox;
      free(del);
      del = aux;
      f->tamanho--;
   }
   return 1;
}
```

2) [1pt] Dos métodos para implementação de filas em array, vistos em sala de aula, o método 3 (Desfragmentando ao atingir o limite), ao inserir um elemento, poderá chamar a função `desfragmentar` (veja o código abaixo). Uma variação desse método seria deslocar os elementos toda vez que fosse realizada uma remoção. Portanto, reescreva o código da operação `sairElemento` de maneira a desfragmentar o array em cada operação de remoção. Além disso, explique se esse código de remoção torna-se menos ou mais eficiente.

```
int entrarElemento(fila *f, elemento e) {
   if ((filaCheia) && (f->tamanho != f->fim+1))
      desfragmentar(f);
   if (!(filaCheia(*f))) {
      f->fim++;
      f->elem[f->fim] = e;
      f->tamanho++;
      return 1;
  } else
     return 0;
}

void desfragmentar(fila *f) {
    int i,x;
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA - IFMA
Curso: Sistemas de Informação
Algoritmos e Estrutura de Dados – Prof. Gentil Cutrim Serra Jr
2ª Avaliação – 16/Novembro/2022

```
    x = f->inicio;
    for (i=0; i<f->tamanho; i++) {
       f->elem[i] = f->elem[x];
       x++;
    }
    f->inicio=0;
    f->fim = f->tamanho-1;
}

int sairElemento(fila *f, elemento *e) {
   if (!(filaVazia(*f))) {
      *e = f->elem[f->inicio];
      f->inicio++;
      f->tamanho--;
      return 1;
   } else
      return 0;
}
```

```
int sairElemento(fila *f, elemento *e) {
   if (!(filaVazia(*f))) {
      *e = f->elem[f->inicio];
      f->inicio++;
      f->tamanho--;
      // desfragmentar
      int x = f->inicio;
      for (int i=0; i<f->tamanho; i++) {
         f->elem[i] = f->elem[x];
         x++;
      }
      f->inicio=0;
      f->fim = f->tamanho-1;
      return 1;
   } else
      return 0;
}
```

3) [2pt] Considere a função iterativa abaixo, que soma os valores positivos de um vetor de números, escreva uma versão recursiva correspondente a esta função. Considere que na chamada da função são passados o vetor **x[]** e a quantidade de elementos do vetor (num).

```
int TTT(int x[], int num) {
   int somatorio=0;
   int n=num-1;
   while (n>=0) {
      if (x[n] > 0)
        somatorio = somatorio + x[n];
      n--;
   }
   return somatorio;
```

}

```
int TTT(int x[], int num) {
  int n = num-1;
  if (n == -1)
     return 0;
  if (x[n] > 0)
     return x[n] + TTT(x, num-1);
  else
     return TTT(x, num-1);
}
```

4) [2pt] Faça uma função para inverter recursivamente a posição dos elementos de uma `fila f` utilizando as funções do TAD Fila (2pt).

```
int inverteFila (tipoFila *f);
```

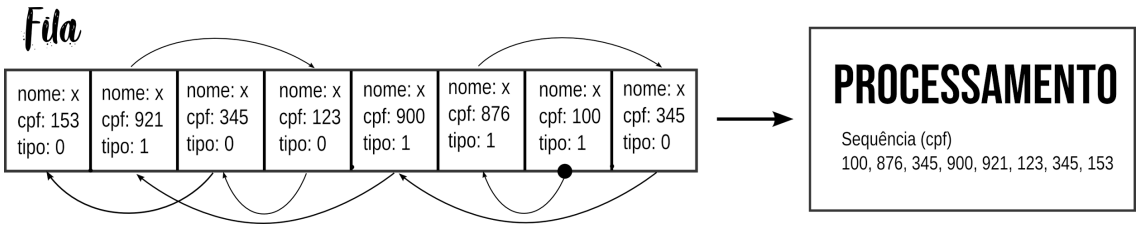
```
int inverteFila (tipoFila *f) {
   elem e;
   sairElemento(f, &e);
   if (!filaVazia(*f)) {
      inverteFila(f);
      entrarElemento(f, e);
   }
   else
      entrarElemento(f, e);
}
```

5) [1pt] Escreva uma função para somar o conteúdo de todos os nós de uma árvore binária, caso o conteúdo do nó seja par (2pt). Considere que em C, um número é par caso `numero % 2 == 0`.

```
int somaParNos(tree t) {
   if (t != NULL)
      int valor = 0;
      if ((t->info % 2) == 0) valor=t->info;
      return (valor + somaParNos(t->esq) + somaParNos(t->dir));
   else
      return 0;
}
```

6) [2pt] O recebimento das declarações de Imposto de Renda Pessoa Física é feito anualmente pela Receita Federal através do programa *ReceitaNet*. Os registros recebidos online por esse programa vão sendo colocados em uma fila (F), de maneira que os primeiros contribuintes a enviar suas declarações têm prioridade no recebimento da

restituição. Contudo, os idosos e doentes crônicos (considerados especiais) têm prioridade ainda maior. Deste modo, todas as declarações são recebidas em uma fila F e ao final do prazo do recebimento, 30 de abril de cada ano, é executado um programa que processa a declaração de **dois contribuintes do tipo prioridade especial** para cada **um do tipo prioridade normal**, nessa ordem, depois repete esse processo até não ter mais contribuintes na fila F. Faça uma função que realiza esse processamento (para simplificar, o processamento irá apenas exibir o nome do contribuinte). Cada elemento da fila deve conter os campos: `nome`, `cpf`, `tipo` (0 para atendimento normal ou 1 para atendimento prioritário).




```
int separar_filas(tipo_fila *f, tipo_fila *f0, tipo_fila *f1) {
   // separa os elementos por tipo de fila
   elemento e;
   while !(filavazia(f) {
      sairElemento(f, &e);
      if (*e.tipo == 0)
         entrarElemento(f0, e);
      else
         entrarElemento(f1, e);
   }
}

int processa (tipo_fila *f) {
   tipo_fila *f0, *f1;
   elemento e;
   separar_filas(tipo_fila f, tipo_fila f0, tipo_fila f1);
   while ((!filavazia(f0)) || (!filavazia(f1)))
      if !(filavazia(f1) {
         sairElemento(f1,&e);
         imprime(*e.nome);
      }
      if !(filavazia(f1) {
         sairElemento(f1,&e);
         imprime(*e.nome);
      }
      if !(filavazia(f0) {
         sairElemento(f0,&e);
         imprime(*e.nome);
      }
   }
}
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA - IFMA
Curso: Sistemas de Informação
Algoritmos e Estrutura de Dados – Prof. Gentil Cutrim Serra Jr
2ª Avaliação – 16/Novembro/2022

Questões extra

---

7) [0,5pt] [IADES - 2019 - BRB - Analista de Tecnologia da Informação - Adaptada] A pilha é uma estrutura de dados que permite a inserção e a remoção desses dados sempre por meio de regras predefinidas. Para que essas operações sejam realizadas, são utilizadas duas funções: push e pop. Com base nessa informação, considere que um programa possua uma pilha p, inicialmente vazia, e que as seguintes operações foram realizadas: PUSH(p, 10); PUSH(p, 5); PUSH(p, 3); PUSH(p, 50); POP(p); PUSH(p, 11); PUSH(p, 9); PUSH(p, 20); POP(p); POP(p). Ao fim da execução desses comandos, quais serão o fundo da pilha e o topo da pilha, respectivamente?
a) 50 e 68.
b) 20 e 58.
c) 11 e 29.
d) 9 e 38.
e) 10 e 11.

8) [0,5pt] [ESAF - 2010 - MPOG - Analista de Planejamento e Orçamento - Tecnologia da Informação - Adaptada] No contexto de estrutura de dados, uma FILA é

a) uma lista do tipo LILO.
b) uma lista do tipo FIFO.
c) um tipo de lista linear em que as operações de inserção e remoção são realizadas na extremidade denominada topo.
d) um tipo de lista linear em que as operações de inserção e remoção são realizadas aleatoriamente.
e) um tipo de lista linear em que as operações de inserção são realizadas em uma extremidade e as operações de remoção são realizadas aleatoriamente.