Bach. em Ciência da Computação — 3º Período — 11/07/2016
Estrutura de Dados — Prof. Hamilton J. Brumatto
Prova 1SC Nome: GABARITO Nº: _____

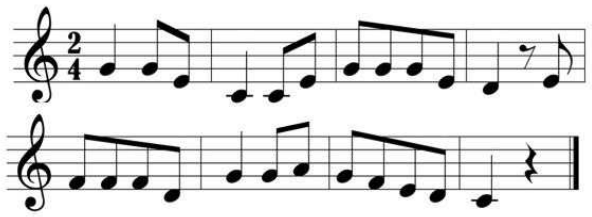
Instruções:
(a) A prova tem a duração de 90 minutos;
(b) A prova é individual sendo proibida qualquer consulta ou o uso de qualquer meio de comunicação;
(c) A interpretação do enunciado é parte integrante da prova;
(d) O total de pontos é proporcional à nota, sendo que a nota equivalente à totalidade de pontos definidos na prova não é menor que 10;
(e) INCLUA O RACIOCÍNIO (ou contas) para chegar à resposta.
(f) Pode indicar o uso de algoritmos de apoio vistos em sala.
(g) Na resposta de uma questão pode ser considerado que a função criada na questão anterior esteja correta.

Questão:	Max	Pontos
Q1	20	
Q2	20	
Q3	20	
Q4	20	
Q5	20	
Total	100	
Nota		

Boa Prova!

O objetivo de Estrutura de Dados é representar elementos do mundo real em um modelo computacional. Aliado à ED temos outras áreas, como IHC (Interface Humano Computador) que usa a modelagem criada para um resultado visual ao usuário. Não vamos aqui explorar IHC, e sim como o problema computacional pode ser modelado em um sistema.

Veja um exemplo, uma partitura musical é representada visualmente na seguinte figura:



Para modelar a partitura encontramos os seguintes elementos:

- nota musical: ela representa a oitava, (depende do instrumento, é o alcance das notas no instrumento, pode ser de 3 a 5 oitavas), a nota em si (são 7 notas), algumas notas aceitam o deslocamento (sustenido ou bemol). A duração (semibreve: 1, mínimima: 1/2, semínima: 1/4, colcheia: 1/8, semicolcheia: 1/16, fusa: 1/32 e semifusa: 1/64) de unidade de tempo, além disto, se a nota for pontuada, ela tem a sua duração acrescida em mais metade de seu tempo.
- pausas: são apenas tempos sem emissão de som. Representa a duração (1/2, ..., 1/64) do tempo sem som.
- compasso: Indica quantos tempos tem um conjunto musical mínimo. Um compasso 3/4 indica que cada unidade de tempo do compasso é um quarto do tempo da semibreve, e ao ser preenchido ocupa 3 unidades de tempo, o compasso.
- melodia: Indica uma sequência de notas e pausas em um compasso, a melodia também precisa de uma clave de referência, que pode ser: sol, fá ou ré, apenas uma referência para, na interface gráfica, representar a altura da posição da nota na pauta.

- música: Um conjunto de melodias, é necessário indicar quantas melodias possui. Todas as melodias em uma música possuem o mesmo compasso e a mesma clave. As músicas possuem outras notações, como intensidade, repetições, ... mas não veremos isto aqui.

Pede-se:

1. (20 pontos) Crie estruturas/classes para representar: nota, pausa, compasso, melodia e música.

```
enum {NADA, SUSTENIDO, BEMOL}; //deslocamento
#define SEMIBREVE 64 // Duração das notas (veja terceira questão)
#define MINIMA 32
#define SEMINIMA 16
#define COLCHEIA 8
#define SEMICOLCHEIA 4
#define FUSA 2
#define SEMIFUSA 1
typedef boolean int;
enum {falso, verdade};
enum {SOL, FA, RE};
enum {NOTA, PAUSA};

typedef struct nota_musical {
   int oitava;
   int nota;
   int deslocamento;
   int duracao;
   boolean pontuada;
} nota_musical_t;

typedef struct pausa_musical {
   int duracao; // posso usar os mesmos nomes para a duração das notas do enum acima.
} pausa_t;

typedef struct compasso {
   int unidade, total;
} compasso_t;

typedef struct melodia_notas {
   int tipo; //nota ou pausa;
   nota_musical_t nota; // ou
   pausa_t pausa;
   struct melodia_notas *prox; // é uma lista de notas/pausas
} melodia_notas_t;

typedef struct melodia {
   int clave;
   compasso_t compasso;
   melodia_notas *notas;
} melodia_t;
```

```
typedef struct musica_melodias {
  melodia_t melodia;
  struct musica_melodias *prox;
} musica_melodias_t;


typedef struct musica {
  int quantidade;
  musica_melodias_t *melodias;
} musica_t;
```

2. (20 pontos) Crie as funções/métodos, que irão definir uma nota, uma pausa, e um compasso com suas características próprias.

```
void criar_nota(nota_musical_t *nt, int oitava, int nota, int desl, int dur, boolean pont) {
    nt− >oitava = oitava;
    nt− >nota = nota;
    nt− >deslocamento = desl;
    nt− >duracao = dur;
    nt− >ponteada = pont;
    return;
};

void criar_pausa(pausa_t *ps, int duracao) {
  ps− >duracao = duracao;
  return;
};

void criar_compasso(compasso_t *cp, int unidade, int total) {
  cp− >unidade = unidade;
  cp− >total = total;
  return;
};
```

3. (20 pontos) Crie as funções/métodos, que irão construir uma melodia inserindo notas e pausas em um compasso.

```
void copiar_nota(nota_musical_t *dest, nota_musical_t *orig) {
    dest− >oitava = orig− >oitava;
    dest− >nota = orig− >nota;
    dest− >deslocamento = orig− >deslocamento;
    dest− >duracao = orig− >duracao;
    dest− >ponteada = orig− >ponteada;
    return;
};

void copiar_pausa(pausa_t *dest, pausa_t *orig) {
  ps− >duracao = duracao;
  return;
};

void copiar_compasso(compasso_t *dest, compasso_t *orig) {
  dest− >unidade = orig− >unidade;
```

```
  dest− >total = orig− >total;
  return;
};

void criar_melodia(melodia_t *ml, int clave, compasso_t compasso) {
  ml− >clave = clave;
  copiar_compasso(ml− >compasso,&compasso);
  notas = NULL;
  return;
};

void inserir_notas(melodia_t *ml, nota_musical_t nt) {
  melodia_notas *pm, *mnt = (melodia_notas *) malloc(sizeof(melodia_notas_t);
  mnt− >tipo = NOTA;
  copiar_nota(&(mnt− >nota),&nt);
  pm = ml− >notas;
  if(pm == NULL) ml− >notas = mnt;
  else {
    while(pm− >prox != NULL) pm = pm− >prox;
    pm− >prox = mnt;
  };
  return;
};

void inserir_pausa(melodia_t *ml, pausa_t ps) {
  melodia_notas *pm, *mnt = (melodia_notas *) malloc(sizeof(melodia_notas_t);
  mnt− >tipo = PAUSA;
  copiar_pausa(&(mnt− >pausa),&ps);
  pm = ml− >notas;
  if(ml == NULL) ml− >notas = mnt;
  else {
    while(pm− >prox != NULL) pm = pm− >prox;
    pm− >prox = mnt;
  };
  return;
};
```

4. (20 pontos) Crie as funções/métodos, que irão verificar se a melodia está construída de forma correta no tempo.
   Vamos contar assim, a menor nota: semifusa, vale 1 ponto. fusa vale 2, semicolcheia 4, colcheia 8, seminima 16, minima 32 e semibreve 64.
   O número de pontos que o compasso deve ter é total/unidade*64. Ou seja, num compasso 2/4 precisamos ter 32 pontos, por exemplo, uma seminima e 2 colcheias completam o compasso.

```
boolean verificar_melodia(melodia_t *ml) {
  int tempo = 0;
  melodia_notas *pm = ml− >notas;
  while(pm != NULL) {
    tempo += pm− >tipo == NOTA ? pm− >nota− >duracao : pm− >pausa− >duracao;
  }
```

return (tempo == (SEMIBREVE * ml− >compasso− >total / ml− >compasso− >unidade))
}

5. (20 pontos) Crie uma função/método que crie uma música a partir das melodias, e a função/método que verifique se a música está montada de forma correta a partir de suas melodias e compassos.
   Só resta agora programa um player que irá tocar a música a partir das melodias.
   Semelhante ao anterior. Criamos a música tendo o total zerado. Criamos o inserir melodias onde na música que serão inseridas recebemos e inserimos as melodias e incrementando o total a cada melodia recebida. A verificação se faz verificando uma a uma melodia com a função da questão anterior. Basta um **e** lógico a cada melodia verificada. Se uma verificar FALSO, então o resultado final será FALSO.