Instituto Superior de Engenharia de Lisboa

Programação II

2021/22 – 2.º semestre letivo

Teste de época de recurso (2022.07.16)

Nota importante: Valoriza-se a escrita de código que inclua comentários esclarecedores da implementação seguida e que contenha indentação legível.

1. [5 valores]

Pretende-se fazer o registo de eventos num calendário, podendo existir eventos diferentes agendados para uma mesma data. Para tal será usado o tipo **`Event`** para fazer o registo de um determinado evento.
A data em que um evento irá decorrer é representada usando o tipo **`Date`**.

```
#define N_MONTHS 12
char *arr_months[] = {"Janeiro", "Fevereiro", ... , "Dezembro"};
```

```
typedef struct {
  short day;
  char  month[MONTH_NAME_MAX];
  short year;
} Date;
```

```
typedef struct {
  Date date;    // Data em que o evento
                // irá decorrer
  char *event; // Designação do evento
} Event;
```

O calendário de eventos será suportado pelo seguinte tipo de dados:

```
typedef struct {
  Event arr[MAX_EVENTS];
  int start_index;
  int end_index;
} Calendar;
```

Em que `arr` corresponde a um *array* de eventos armazenados consecutivamente entre os índices `start_index` e `end_index`, incluindo estas posições. Assim, o primeiro evento armazenado no *array* pode não se encontrar na posição índice zero. Assuma que `end_index` é igual ou superior a `start_index`. Caso não existam eventos armazenados no calendário `start_index` tem o valor -1.

a) [1,0] Escreva a função
`void load_event(Event *ev, int day, int month, int year, char *event_name);`
que recebe uma referência para uma estrutura do tipo `Event` (previamente criada) e faz o seu preenchimento a partir dos dados enviados à função. A *string* `event_name` deve ser reproduzida em alojamento dinâmico.

b) [1,0] Escreva a função
`int month_ok(Date date, int *out_month);`
que recebe uma estrutura do tipo **`Date`** e devolve um valor lógico que indica se o nome do mês registado em *date* existe no *array* `arr_months` que contém **`N_MONTHS`** *strings*. Se o nome existir, a função coloca na variável apontada por `out_month` o número do mês correspondente (1..12); caso contrário, coloca 0.

c) [1,0] Escreva a função
`int event_compare(const void *ev1, const void *ev2);`
que compara dois eventos, retornando um valor negativo, zero ou um valor positivo, usando o seguinte critério, pela ordem a seguir indicada:
   1. Datas crescentes;
   2. Datas iguais, desempatam com a descrição do evento de forma crescente.

NOTA: Supõe-se que as datas presentes nos registos são válidas.

d) [2,0] Escreva a função
```
Event *find_event(Calendar *cal, int day, int month, int year,
                  Char *event_name);
```
que verifica se existe um evento com as características enviadas à função. Para tal, deverá:
- Ordenar a lista de eventos recorrendo à função **`qsort`** da biblioteca *standard* da linguagem;
- Realizar uma pesquisa binária ao *array* já ordenado, usando para tal a função **`bsearch`** da biblioteca *standard* da linguagem.

A função deverá devolver o endereço do elemento na lista ou NULL caso não exista.

2. [5 valores]

Considere as seguintes estruturas, especificadas para a realização das Séries de Exercícios 3 e 4, representadas na caixa seguinte. Recorde o parágrafo do enunciado de uma das séries sobre uma das estruturas: "No (...) tipo **`DinRef_t`** o campo `refs` é um ponteiro, destinado a apontar para um *array* de ponteiros, de modo a permitir o seu alojamento e realojamento dinâmico, para se adaptar automaticamente à quantidade de *tags* existente. O campo `space` serve para controlar a quantidade de elementos alojados, de modo a permitir o realojamento por blocos com o objetivo de evitar o custo de um realojamento por cada elemento adicionado." Antes de iniciar a realização deste grupo, leia o enunciado de todas as alíneas que o compõem.

```
1  typedef struct{
2    char title[MAX_TIT + 1];
3    char artist[MAX_ART + 1];
4    char album[MAX_ALB + 1];
5    short year;
6    char comment[MAX_COM + 1];
7    char track;
8    char genre;
9  } MP3Tag_t;
10
11 typedef struct{
12   int space;       // Quantidade de elementos alojados no campo refs.
13   int count;       // Quantidade de elementos preenchidos no campo refs.
14   MP3Tag_t **refs; // Ponteiro para array de ponteiros para MP3Tag_t. O
15                    // array e as estruturas MP3Tag_t têm aloj. dinâmico.
16 } DinRef_t;
```

a) [1,0] Pretende-se o desenvolvimento da função

```
int ref_add_no_rep(DinRef_t *ref, MP3Tag_t *tag);
```

que adiciona ao descritor, indicado por `ref`, o ponteiro para a estrutura indicada por `tag`, previamente alojada e preenchida, garantindo que não há repetições, ou seja, garantindo que o *array* contido em `ref` não referencia a mesma `tag` mais que uma vez. A função retorna 1 se houver adição; retorna 0 se não houver adição por já existir; retorna -1 se não houver adição por não haver memória dinâmica disponível no *heap*.

b) [1,0] Pretende-se o desenvolvimento da função

```
int delete_refs(DinRef_t *dr, short year);
```

que admite que o "*array*" de referências (passado no 1.º parâmetro) está ordenado ascendentemente por `year` e remove deste "*array*" todas as referências a *tags* cujo ano seja igual ao parâmetro `year`, mantendo o "*array*" ordenado (por `year`). A função retorna o número de *tags* removidas, atualizando o membro `count` convenientemente mas mantendo o valor registado no membro `space` (atuando de acordo com esta premissa). Não é permitida a utilização da função `qsort` da biblioteca normalizada.

c) [1,0] Pretende-se o desenvolvimento da função

```
void ref_scan(DinRef_t *dr, void(*action)(MP3Tag_t *tag, void *context),
              void *context);
```

que percorre o "*array*" de referências (passado no 1.º parâmetro), aplicando a função passada em `action` a cada uma das *tags* referenciadas, passando-lhe a referência para a *tag* e o parâmetro `context` que recebeu.

d) [2,0] Como exemplo de utilização da função anterior, pretende-se o desenvolvimento da função

```
void print_year(DinRef_t *dr, short year);
```

que, recorrendo à função `ref_scan`, afixa em *standard output* a informação contida nas *tags* cujo ano seja igual a `year` (crie uma linha por *tag*, usando a formatação que entender dentro da linha).

3. [4 valores]

Pretende-se realizar um programa para gestão de mensagens ativadas por tempo (lembretes), armazenadas numa lista ligada com nós do tipo **`MsgNode`**.

```
typedef struct msgNode {
    struct msgNode *next;   // ligação em lista
    long time;              // instante de tempo programado
    char *message;          // mensagem a apresentar; string alojada dinamicamente
} MsgNode;
```

As listas ligadas que representam os conjuntos são ordenadas por valor crescente do campo `time`. Na implementação das funções pretendidas, pode escrever funções auxiliares, se considerar conveniente. Valoriza-se a eficiência.

a) [1,0] Escreva a função

```
void msgInsert( MsgNode **headptr, long time, char *msg  );
```

que insere um elemento com o valor do parâmetro `time` e réplica, alojada dinamicamente, da *string* `msg`. . A função deve manter atualizado o ponteiro cabeça de lista, indicado pelo parâmetro `headptr`.

b) [1,0] Escreva a função

```
int msgCount( MsgNode *head );
```

que retorna a quantidade de mensagens armazenadas na lista indicada por `head`.

c) [2,0] Escreva a função

```
void msgProcess( MsgNode **headptr, long curtime );
```

que verifica se há, na lista acessível através de `headptr`, elementos com tempo programado igual ou inferior a `curtime`. Se houver, apresenta em *standard output* as respetivas mensagens e elimina esses elementos, libertando toda a memória de alojamento dinâmico que eles ocupam. A função deve manter atualizado o ponteiro cabeça de lista, indicado pelo parâmetro `headptr`.

4. [3 valores]

Considere uma árvore binária de pesquisa com dados genéricos, formada por nós com o tipo **`TNode`** seguinte.

```
typedef struct tNode{
    struct tNode *left, *right;    // ponteiros de ligação na árvore
    void *data;                    // ponteiro para o bloco de dados
} TNode;
```

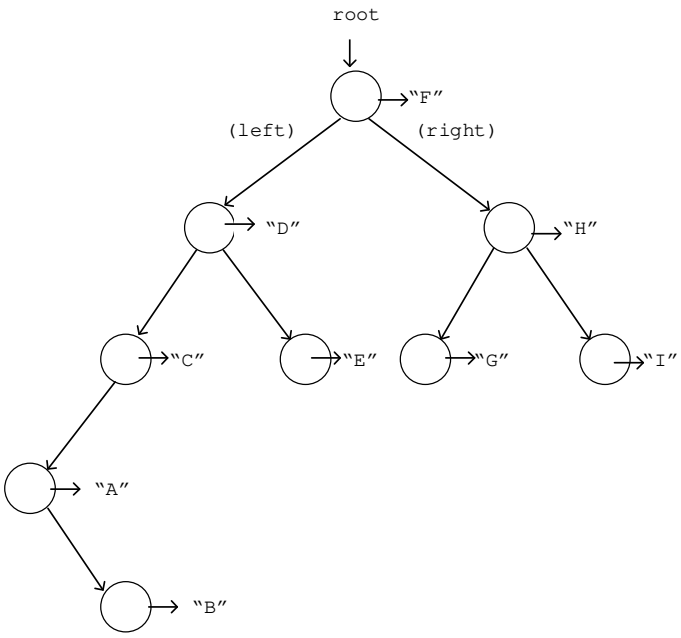
Pretende-se a análise do comportamento recursivo de uma função que avalia o estado de balanceamento de árvores binárias de pesquisa. Uma árvore considera-se balanceada se, em todos os seus elementos, a diferença de alturas das subárvores é 0 ou 1; considera-se desbalanceada se, em algum nó, aquela diferença é superior a 1.

A função `tUnbalanced` seguinte verifica o estado de balanceamento de uma árvore, retornando 1 no caso de esta se encontrar desbalanceada; tem também um parâmetro de saída (`h`, *height*) que serve para afetar uma variável, passada por endereço, com a altura da árvore observada.

```
int tUnbalanced( TNode *r, int *h ){
    if( r == NULL ){
        *h = 0;
        return 0;
    }
    int  h1, h2, res1, res2;
    res1 = tUnbalanced( r->left, &h1 );
    res2 = tUnbalanced( r->right, &h2 );
    *h = 1 + h1 > h2 ? h1 : h2;
    int dif = abs( h1 - h2 );
    if( dif > 1 || res1 || res2 )
        return 1;
    return 0;
}
```

Admita que uma árvore binária de pesquisa, com nós do tipo referido, é usada para armazenar *strings*, apontadas pelos campos `data` dos seus nós, com ordenação alfabética crescente da esquerda para a direita.

A árvore já foi construída e tem a topologia representada na figura ao lado.



a) [1,0] A função `tUnbalanced` é chamada recebendo em `r` a raiz da árvore representada na figura.

   Tendo em conta o algoritmo recursivo, indique a quantidade total de vezes que a função `tUnbalanced` é chamada, incluindo a primeira. Identifique os nós que são observados nesta atividade da função.

b) [1,0] Desenhe uma nova topologia da árvore, com os mesmos nós reorganizados de modo a ficar balanceada.

c) [1,0] Considere agora que a função `tUnbalanced` é chamada recebendo em `r` a raiz da árvore reorganizada na alínea anterior. Indique, para este caso, a quantidade total de vezes que a função `tUnbalanced` é chamada, incluindo a primeira. Identifique os nós observados nesta atividade da função.

5. [3 valores]

Considere as definições seguintes, relativas à representação de conjuntos de valores numéricos.

```
#define MAX_SET_VAL ( sizeof( long ) * CHAR_BIT - 1 )
typedef unsigned long Set;
```

O código para manipulação dos conjuntos é implementado em funções definidas nos módulos seguintes:

Nomes dos módulos	Funções definidas nos módulos

setempty.c	`Set setEmpty( void ){`
	`  return 0;`
	`}`

setsingle.c	`Set setSingle( int v ){`
	`  return 1UL << v;`
	`}`

setintersect.c	`Set setIntersect( Set s1, Set s2 ){`
	`  return s1 & s2;`
	`}`

setreunion.c	`Set setReunion( Set s1, Set s2 ){`
	`  return s1 | s2;`
	`}`

setcomplem.c	`Set setComplem( Set s ){`
	`  return ~s;`
	`}`

setinsert.c	`Set setInsert( Set s, int v ){`
	`  return setReunion( s, setSingle( v ) );`
	`}`

setremove.c	`Set setRemove( Set s, int v ){`
	`  return setIntersect( s, setComplem( setSingle( v ) ) );`
	`}`

setverify.c	`int setVerify( Set s, int v ){`
	`  return setIntersect( s, setSingle( v ) ) != setEmpty();`
	`}`

setprint.c	`void printInt(int v ){`
	`  printf("%d\n", v );`
	`}`
	`void setPrint( Set s ){`
	`  for( int i = 0; i <= MAX_SET_VAL; ++i ){`
	`  if( setVerify( s, i ) ){`
	`  printInt( i );`
	`  }`
	`  }`
	`}`

Admita que é criado e usado no processo de compilação um *header file*, **`set.h`**, contendo as definições necessárias e as assinaturas das funções públicas de todos os módulos fonte (**`.c`**) referidos, e que poderão ser desenvolvidos diversos programas de aplicação, os quais poderão utilizar quaisquer das funções referidas. Considere que os módulos fonte são compilados com o comando "**`gcc -c *.c`**".

a) [1,0] Indique se o atributo `static` pode ser usado em alguma(s) das funções anteriormente referidas. Justifique, referindo as condições para que este atributo seja ou não aplicável a qualquer função.

b) [1,0] Considerando que aplica a conclusão da alínea a), apresente a lista de símbolos produzida por "**`nm setprint.o`**" e a respetiva classificação: **`T`**, *public text*; **`t`**, *private text*; **`U`**, *undefined*. Por cada um dos símbolos *undefined*, indique qual o módulo que o resolve ou se é resolvido pela biblioteca.

c) [1,0] Suponha que é desenvolvido um módulo de aplicação com o nome "**`demo_set.c`**", o qual chama apenas as funções **`setEmpty`**, **`setInsert`** e **`setPrint`**. Escreva um *makefile* que produza, de forma eficiente, o executável correspondente, com o nome "**`demo_set`**", a partir dos módulos fonte (**`.c`**) estritamente necessários.