utad Universidade de Trás-os-Montes e Alto Douro

Escola de Ciências e Tecnologia. Departamento de Engenharias

		%
1	7.5	
2	7	
3	5.5	

Eng. Informática

unidade curricular: **Algoritmia** ano lectivo: **2015 / 2016**

Teste Prático 02. **19.Mai.2016** duração: **55 min** (+5 min tolerância)

(sem consulta)

Número: _____ Nome: _____

O meu amigo Xavier guardou os dados da sua árvore genealógica (até aos bisavós) numa árvore binária em que ele é a raiz, os seus progenitores são os seus filhos na árvore (masculino à esquerda e feminino à direita) e assim sucessivamente (os filhos do seu pai são o avô paterno à esquerda e a avó paterna à direita, etc.).
Considere o seguinte programa em linguagem C (implementação dessa árvore binária):

```
#define DATA(node)  ((node)->pData)
#define LEFT(node)  ((node)->pLeft)
#define RIGHT(node) ((node)->pRight)
typedef enum _STATUS {OK,ERROR} STATUS;
typedef enum _BOOLEAN {FALSE=0,TRUE=1} BOOLEAN;

// Definição da estrutura PERSON
typedef struct _PERSON {
   char name[30];
   int age;
   BOOLEAN deceased;
} PERSON;

// Definição da estrutura BT_NODE
typedef struct _BT_NODE {
   void *pData;
   struct _BT_NODE *pLeft;
   struct _BT_NODE *pRight;
} BT_NODE;

typedef BT_NODE *BT;

//Declaração das funções
BT_NODE *initNode(void *, BT_NODE *, BT_NODE *);
BT_NODE *createNewBTNode(void *);
BT createBT(PERSON *, int, int);
BOOLEAN emptyBTree(BT);
BOOLEAN isLeaf(BT_NODE *);
void printPersonBTree(BT);

void changeExpenses(BT, float, float);
void changeStaff(BT);
BT transformBTree(BT);

#define NMAX 15

int main(int argc, char *argv[])
{
   BT T = NULL;

   PERSON v[]={"Xavier",45,FALSE,
               "Jorge",74,FALSE,"Luisa",69,FALSE,
               "Raul",58,TRUE,"Joaquina",72,TRUE,"Marco",77,TRUE,"Teresa",91,FALSE,
               "Antero",70,TRUE,"Maria",86,TRUE,"Carlos",81,TRUE,"Sandra",36,TRUE,
               "Duarte",72,TRUE,"Ilda",84,TRUE,"Alfredo",55,TRUE,"Francisca",77,TRUE};

   T = createBT(v,0,NMAX); printPersonBTree(T);
   ...
   return EXIT_SUCCESS;
}
```

Considere ainda as seguintes funções:

```
// Função que verifica se uma árvore binária está vazia

BOOLEAN emptyBTree(BT T)
{
   return (T==NULL)? TRUE : FALSE;
}

// Função que verifica se um nó é uma folha

BOOLEAN isLeaf(BT_NODE *pT)
{
   return ((LEFT(pT)==NULL) && (RIGHT(pT)==NULL))? TRUE : FALSE;
}


// Função que cria um nó

BT_NODE *createNewBTNode(void *pData)
{
   BT_NODE *pTemp;

   if ( (pTemp = (BT_NODE *)malloc(sizeof(BT_NODE))) != NULL ) {
      DATA(pTemp) = pData; LEFT(pTemp) = RIGHT(pTemp) = NULL;
   }

   return pTemp;
}


// Funções que criam a BST

BT createBT(PERSON v[], int i, int size)
{
   if( i >= size ) return(NULL);
   else return(initNode(&v[i],createBT(v,2*i+1,size),createBT(v,2*i+2,size)));
}

BT_NODE *initNode(void *pData, BT_NODE *n1, BT_NODE *n2)
{
   BT_NODE *pTemp = NULL;

   pTemp = createNewBTNode(pData);
   LEFT(pTemp) = n1;
   RIGHT(pTemp) = n2;

   return(pTemp);
}


// Função que apresenta a árvore binária (de elementos de tipo PERSON)

void printPersonBTree(BT T)
{
   if(emptyBTree(T)==TRUE) return;

   printPersonBTree(LEFT(T));
   printf("\nnome : %s - idade : %d (%s)", ((PERSON *)DATA(T))->name,
         ((PERSON *)DATA(T))->age,((PERSON *)DATA(T))->deceased ? "falecido": "vivo");
   printPersonBTree(RIGHT(T));

   return;
}
```

01 Implemente a função **`printParents`**, que apresente os pais de determinada pessoa, cujo nome (admita que não há nomes repetidos) deve ser passado como argumento. Se não existir essa informação deve dizê-lo.

```
void printParents(BT T, char *name)
{




}
```

02 Implemente a função **`maxAgeGreatGreatParents`**, que retorne o número de anos do bisavô ou bisavó que mais viveu.

```
int maxAgeGreatGreatParents(BT T)
{




}
```

03 Implemente uma função **`maxAgeSide`** que determine qual o ramo da família (paterno ou materno) tem a pessou que viveu mais anos. Se for do lado do pai o resultado deve ser zero, senão deve ser um. Crie as funções auxiliares que entender necessárias.

```
int maxAgeSide(BT T)
{









}
```