2º Teste de Princípios de Programação Procedimental
17 de junho de 2022
Duração 2 horas

Departamento de Engenharia Informática - FCTUC - Universidade de Coimbra
Licenciatura em Engenharia Informática

Este teste é com consulta, mas apenas de recursos em papel. É proibido o uso de qualquer aparelho eletrónico (smartwatches, telemóveis, tablets, computadores, etc.). Quem tiver trazido esses aparelhos deve desligá-los e colocá-los no chão, não podendo mexer neles durante a prova. A violação desta regra, assim como qualquer tentativa de comunicar com qualquer pessoa para além dos docentes que estão a vigiá-la, pode levar à anulação da prova.

Nota: Preencha com o nome e nº completos e responda nos espaços reservados para o efeito. **Duração total**: 2h.

Nome:_____ Número:_____

1. Apresente na consola a saída resultante da execução do seguinte excerto de código.

Código:

```
#define MAX_SIZE 3

int step = 0;

char text[10];

char table[] = { 'D', 'D', 'F', 'P', 'F',
'P', 'O', 'K', 'Z'};

strcpy(text, "ABCDEF");

printf("%s\n", text);

text[MAX_SIZE] = '\0';

printf("%s\n", text);

do {
 for (int i=0; i < strlen(text); i++){
    text[i] = table[text[i] - 'A'];
 }
 ++step;

 printf("%d->%s\n", step, text);

} while (strcmp(text, "PPP") != 0);

strcat(text, "2022");

printf("%s\n", text);
```

Consola:

ABCDEF
ABC
1->DDF
2->PPP
PPP2022

Nome:_____ Número:_____

2. Complete o código da função **expressao_valida** apresentada em baixo. A função recebe uma string (ponteiro para char) com uma expressão aritmética, devolvendo 1 se a expressão estiver correta em termos de balanceamento de parêntesis curvos e retos, e 0 caso contrário. A função não verifica outros erros. Exemplos de expressões consideradas corretas:

   "(5)", "1 + 1", "(1 + 1)", "((1 + 1))", "vec[0]", "vec[(1 + 7) * 2]", "vec[vec2[0] * vec3[0]]", "(2 + 5 * vec[n - 1]) + 2"

   Exemplos de expressões consideradas incorretas:
   "(1 + 1", "1 + 1)", "vec[5", "vec[vec2[0] * vec3[0]", "vec[5)]"

   A função **expressao_valida** usa uma pilha de char auxiliar para determinar a correção da expressão. As estruturas de dados assim como as funções auxiliares são as seguintes:

```
typedef struct no_pilha_char {
    char c;
    struct no_pilha_char *anterior;
} no_pilha_char;

typedef struct {
    no_pilha_char *topo;
} pilha_char;

void init_pilha_char(pilha_char
*pilha) {
    pilha->topo = NULL;
}
```

// Cria um novo nó, em memória dinâmica, com o char c e coloca-o no topo da pilha

```
void push_pilha_char(pilha_char
*pilha, char c);
```

// Devolve o char do nó do topo da pilha, removendo este nó da pilha e libertando a memória dinâmica correspondente. É seguro chamar a função mesmo que a pilha esteja vazia, devolvendo neste caso o char '\0'.

```
char pop_pilha_char(pilha_char
*pilha);
```

// Elimina da memória dinâmica todos os nós da pilha e coloca o ponteiro para o topo da pilha a NULL. É seguro chamar a função mesmo que a pilha esteja vazia.

```
void delete_pilha_char(pilha_char
*pilha);
```

```
int expressao_valida(char *str) {
    pilha_char pilha;
    init_pilha_char(&pilha);
    int n = strlen(str);
    for (int i = 0; i < n; ++i) {
        if (str[i] == '(' || str[i] == '[')
            push_pilha_char(&pilha, str[i]);
        else if (str[i] == ')' || str[i] == ']') {
            char c = pop_pilha_char(&pilha);
            if (str[i] == ')' && c != '(' || str[i] == ']' && c != '[') {
                delete_pilha_char(&pilha);
                return 0;
            }
        }
    }
    if (pilha.topo == NULL)
        return 1;
    delete_pilha_char(&pilha);
    return 0;}
```

Nome:_____ Número:_____

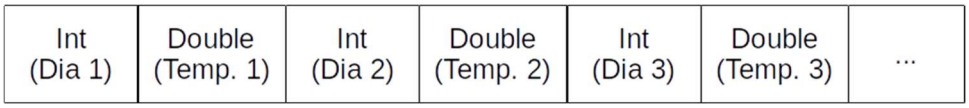
3. Considere o seguinte código para representar uma fila de registos de temperatura para determinados dias do ano:

```
struct registo {
  int dia;
  double temperatura;
  struct registo *prox;
};
```

```
struct fila {
  struct registo *inicio;
  struct registo *fim;
};
```

Os apontadores `inicio` e `fim` estão a NULL se a fila estiver vazia.

Considere ainda que tem um ficheiro binário com os valores das temperaturas guardados, com o seguinte formato:



Ou seja um conjunto de pares de valores, em que o primeiro é um `int`, que corresponde ao dia do ano, e o segundo um `double` que corresponde à temperatura nesse dia.

Tendo em conta a estrutura de dados e a estrutura do ficheiro binário descritas, escreva uma função que leia os valores de temperatura do ficheiro para uma fila por ordem, de forma a que o primeiro par de valores do ficheiro fique no início da fila, e o último no fim. A função deve devolver o número de pares lidos do ficheiro para a fila, e ter o seguinte cabeçalho:

`int le_registos(const char * nome_ficheiro, struct fila * f);`

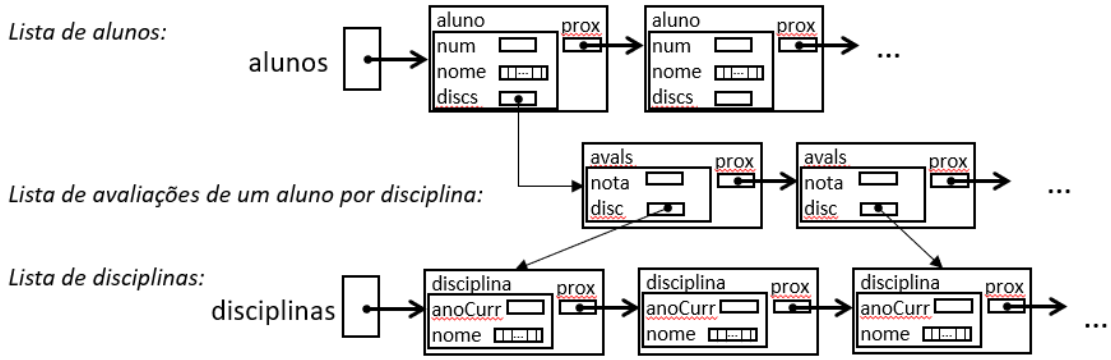
Nota: não devem ser escritas funções nem estruturas auxiliares.

```
int le_registos(const char * nome_ficheiro, struct fila * f) {
  if (f == NULL || nome_ficheiro == NULL) return 0;
  FILE *fp = fopen(nome_ficheiro, "rb");
  if (fp == NULL) return 0;
  int count = 0;
  int dia;
  double temperatura;
  while (1) {
    if(fread(&dia, sizeof(int), 1, fp) != 1) break;
    if(fread(&temperatura, sizeof(double), 1, fp) != 1) break;
    struct registo *r = malloc(sizeof(struct registo));
    if (r == NULL) break;
    r->dia = dia;
    r->temperatura = temperatura;
    r->prox = NULL;
    if (f->fim == NULL) {
      f->inicio = r;
      f->fim = r;
    } else {
      f->fim->prox = r;
```

```
      f->fim = r;
    }
    count += 1;
  }
  fclose(fp);
  return count;
}
```

Nome:_____ Número:_____

4. Considere um programa que gere as avaliações de cada aluno, em cada disciplina, com as seguintes estruturas:



As estruturas e tipos de dados implementados neste programa são os seguintes:

```
#define MAX 50

// Lista de disciplinas
typedef struct disciplina {
  int anoCurr;
  char nome[MAX];
}tipoDisciplina;

typedef struct noDisciplina{
  tipoDisciplina disciplina;
  struct noDisciplina * prox;
}tipoNoDisciplina;

// Lista de Avaliações de um aluno
por disciplina
typedef struct avalDisciplina {
  tipoNoDisciplina * disc;
  int nota;
} tipoAvalDisciplina;

typedef struct noAvaliacoes {
  tipoAvalDisciplina avals;
  struct noAvaliacoes * prox;
} tipoNoAvaliacoes;

// Lista de alunos
typedef struct aluno {
  int num;
  char nome [MAX];
  tipoNoAvaliacoes * discs;
} tipoAluno;

typedef struct noAluno {
  tipoAluno aluno;
  struct noAluno * prox;
} tipoNoAluno;
```

Considere o seguinte código:

```
void listaNotasAlunos (tipoNoAluno * alunos);

int main() {
    tipoNoDisciplina * disciplinas;
    tipoNoAluno * alunos;
    //...
    listaNotasAlunos (alunos);
    return 0;
}
```

Implemente **a função `void listaNotasAlunos (tipoNoAluno * alunos)`** que deve listar (apresentar no ecrã) todos os alunos (apenas o nome do aluno) e, para cada aluno, a disciplina onde obteve a melhor nota. Caso a lista de notas se encontre vazia (NULL), deverá escrever "Sem avaliação."

Não serão aceites funções adicionais (para além da solicitada). **v.s.f.f.-->**

```
void listaNotasAlunos (tipoNoAluno * alunos){

    tipoNoAluno * al = alunos;
    tipoNoAvaliacoes * disc, * maior;

    printf("Avaliacao final de cada aluno por disciplina:\n");
    while (al != NULL){
        printf("Aluno: %d %s\n", al->aluno.nome);
        disc = al->aluno.discs;
        if (disc == NULL)
            printf("Sem avaliação.\n");
        else {
            maior = disc;
            while (disc != NULL) {
                if (disc->avals.nota > maior->avals.nota)
                    maior = disc;
                disc = disc->prox;
            }
            printf("%s: %d\n", maior->avals.disc->disciplina.nome,
disc->avals.nota);
        }
        al = al->prox;
    }
}
```