1. [3] Considere os ficheiros fonte:

a.c
```
#include "c.h"
extern int a;
static int *b=&a;
char c=50;
void g() {
  f h = {20,30};
  k(&a);
  printf("%d %d %d %d %d\n",
    a, *b, ++c, h.d, h.e );
}
static void k(int *p) {*p+=5;}
```

b.c
```
#include "c.h"
int a=40;
int h[] = {21,31};
static int *b=h;
char c;
int main() {
  printf("%d %d %d %d %d\n",
    a, *b, c++, h[0], *(h+1));
  g();
  return 0;
}
```

c.h
```
#include <stdio.h>

typedef struct _s {
  int d; int e;
} f;

void g();

static void k(int*);
```

a) [1] O comando **`gcc -Wall -c b.c`** reporta um aviso relativamente à função **`k()`**, mas se for removida a última linha do ficheiro **`c.h`** passa a dar erro quando se compila o módulo **`a.c`** . Porquê?

b) [1] Que símbolos públicos, definidos e indefinidos, constam no módulo **`a.o`**?

c) [1] Considerando o executável produzido com **`gcc a.c b.c`**, indique o *output* da sua execução.

2. [1] Discuta a seguinte afirmação: "No caso das bibliotecas de carregamento dinâmico carregadas em tempo de execução com *dlopen*, as únicas funções que poderão ser executadas são aquelas cujo endereço for obtido com *dlsym*."

3. [9,5] Na realização de um programa para gerir os docentes da área departamental, considere os ficheiros `Teacher.h` e `Teacher.c`. Cada docente tem o número (1..2000 ou 9001..9199), o nome e o email. A informação de todos os docentes é inicialmente carregada a partir de um ficheiro de texto em que cada linha tem estes três campos separados pelo carácter '|'. No índice `idx` do *array* `teachers` fica um ponteiro para o docente com o número `num` em que `num==idx` ou `num-7000==idx` se `num>9000`.

```
#include <stdio.h>

typedef struct _teacher {
  unsigned num; /* 1..2000 or 9001..9199 */
  char *name;
  char *email;
} Teacher;

Teacher * newTeacher(unsigned num,
    char *name, char *email);

void deleteTeacher(Teacher *t);
Teacher * readTeacher(FILE *f);

#define FIELDS 3
#define MAX_TEACHERS 2200
extern Teacher * teachers[MAX_TEACHERS];

unsigned getIdx(unsigned num);
void insertTeacher(Teacher *t);
Teacher * getTeacher(unsigned num);
void printAll();

int split(char *line, int max,
    char *flds[], char sep);
```

```
#include "Teacher.h"
#include <string.h>

unsigned getIdx(unsigned num) {
  if (num>=9200 || num>2000 && num<=9000) return 0;
  return num>9000 ? num-7000 : num;
}
Teacher * getTeacher(unsigned num) {
  return teachers[getIdx(num)];
}
void deleteTeacher(Teacher *t) { free(t); }
void printAll() {
  Teacher **ts;
  for(ts=trachers ; ts < teachers+MAX_TEACHERS ; ++ts)
    if ( *ts ) printTeacher(*ts);
}
#define MAX_LINE 512
static char line[MAX_LINE];

Teacher * readTeacher(FILE *f) {
  char *flds[TEACHER_FIELDS];
  unsigned n;
  if ( !fgets(line,MAX_LINE,f) ) return NULL;
  line[ strlen(line)-1 ] = 0; /* retira '\n' */
  if (split(line,FIELDS,flds,'|') < FIELDS) return NULL;
  sscanf(flds[0],"%d",&n);
  return newTeacher(n,flds[1],flds[2]);
}
...

int main() { ... }
```

(continua)

a) [1] A função `getTeacher` retorna `NULL` para um número de docente inválido? Justifique a resposta.

b) [2] Implemente em IA-32 a função `insertTeacher` que dado um docente chama `getIdx` e afecta a entrada correspondente no *array* `teachers` com o ponteiro para o docente, se o seu número for válido.

c) [2] Implemente em IA-32 a função `printAll`

d) [2] Implemente em C a função `newTeacher`, alojando dinamicamente um único bloco de memória para o espaço total necessário (estrutura + caracteres do nome + caracteres do email).

e) [2,5] Implemente em C a função `split`, que dada a string `line` com campos separados pelo carácter `sep` e um array `flds` com `max` ponteiros para `char`, substitui os caracteres `sep` por 0 e preenche o *array* `flds` com ponteiros para o primeiro carácter de cada campo. Retorna o número de campos encontrados (inferior ou igual a `max`). Valorizam-se soluções que façam uma só passagem em `line`.

4. [1,5] Considerando um sistema com dimensões de tipos iguais às do ambiente de desenvolvimento utilizado em PSC e com uma *cache* de dados de 64KB, *4-way set-associative*, com 1024 *sets* por via, qual o número máximo de linhas de *cache* necessário para representar uma instância da seguinte estrutura? Justifique detalhadamente a sua resposta.

```
struct DataRecord {
    int   record_id;
    float df_factor;
    float gf_factor;
    char  description[48];
};
```

5. [5] A classe abstracta *StrValidator* valida uma *string*, invocando o método abstracto *isCharOk* para cada um dos seus caracteres. As classes concretas derivadas de *StrValidator* implementam critérios de validação específicos em *isCharOk*. Por exemplo, na classe *NoSpaces* confirma-se que não há espaços na *string*, enquanto que *InLimits* verifica se a gama de caracteres está limitada entre dois valores indicados na construção. Apresente uma versão em C, igualmente extensível, do seguinte código Java:

```
abstract class StrValidator {
  abstract boolean isCharOk(char c);
  final boolean isValid(String str)  {
    for (int i = 0; i < str.length(); ++i)
      if (!isCharOk(str.charAt(i))) return false;
    return true;
  }
}
class NoSpaces extends StrValidator {
  boolean isCharOk(char c) { return c != ' '; }
}
class InLimits extends StrValidator {
  char c1, c2;
  boolean isCharOk(char c) { return c >= c1 && c <= c2; }
  InLimits(char low, char high) { c1 = low; c2 = high; }
}
class TestValidator {
  static boolean vall(String[] args, StrValidator v) {
    for (String s : args) if (!v.isValid(s)) return false;
    return true;
  }
  public static void main(String[] args) {
    System.out.println( vall(args, new NoSpaces())        ? "OK" : "NOK");
    System.out.println( vall(args, new InLimits('A','Z')) ? "OK" : "NOK");
  }
}
```

Duração: 2 horas e 30 minutos

Bom teste!