1. [2] Considere o seguinte função `main()`:

```
int main() {
  char *t = "ISEL";
  int a[]= {20,10,5,7,9};
  PRINT_ARRAY(t,"%c",4);
  PRINT_ARRAY(a,"%d,",5);
  return 0;
} /* output: ISEL20,10,5,7,9 */
```

a) [1] Defina a macro `PRINT_ARRAY` para apresentar no standard output os elementos do *array* (de tipos primitivos) indicado como primeiro parâmetro segundo a formatação especificada no segundo parâmetro. O terceiro parâmetro indica a dimensão do *array*.

b) [1] Defina uma função `PRINT_ARRAY` com os mesmos parâmetros e o mesmo objectivo, ou caso entenda que não é possível, justifique.

2. [3] Considere os ficheiros fonte:

fa.c
```
#include <stdio.h>
struct X { int a; char b; };
static struct X x = {10,'A'};
int y = 1;
int main() {
  char c=f(y+x.b);
  putchar(c); return 0;
}
```

fb.c
```
typedef unsigned int X;
X x = 2;
extern int y;
static int * g(int b) {
  int res=1;
  res+=b+y; return &res;
}
int f(int a) { return ++a; }
```

a) [1] Indique um *warning*/aviso que resulte de cada uma das compilações com a opção `-Wall`.

b) [1] Quais são os símbolos públicos/globais que constam em cada um dos módulos compilados? Para cada símbolo indique se é T-*text*, D-*data* ou U-*undefined*.

c) [1] É possível ligar os módulos compilados? Se sim, indique o output do programa, se não, justifique.

3. [10] Na realização de um programa para gerir as estadias de um hotel, considere os ficheiro `Hotel.h` e `Hotel.c`. Cada estadia armazena o nome do cliente e os números do quarto e do andar. A variável global `stays` é um *array* bidimensional de ponteiros para estadias que estão a `NULL` se o respectivo quarto está livre. As estadias são alojadas dinamicamente por cada entrada de cliente no hotel.

```
#include <stdio.h>
#include <malloc.h>
#include <string.h>

#define ROOMS ... /* alínea a) */
#define FLOORS  4

typedef struct _stay {
  char *name;    /* Nome do cliente */
  int floor;     /* num. do andar 0..N */
  int room;     /* num. do quarto 0..N */
} Stay; /* Estadia */

extern Stay * stays[FLOORS][ROOMS];

void enter(char *name, int floor, int room);
void leave(int floor, int room);
Stay * stay(int floor, int room);
Stay * findStay(char *name);
Stay * procStays(int (*fx)(Stay *));
void setStay(Stay *s, char *n, int f, int r);
void leaveAll();
```

```
#include "Hotel.h"

Stay * stays[FLOORS][ROOMS];

Stay * stay(int f, int r) { return stays[f][r]; }

Stay *procStays(int (*fx)(Stay *)) {
  Stay *s, **ss= (Stay**)stays;
  for( ; ss < ((Stay**)stays)+FLOORS*ROOMS ; ++ss)
    if ((s=*ss) && fx(s)) return s;
  return 0;
}

static int freeStay(Stay *s)
{free(s->name); free(s); return 0;}
void leaveAll() { procStays(freeStay); }

void enter(char *name, int floor, int room) { ... }
void leave(int floor, int room)
{ freeStay(stays[floor][room]); stays[floor][room]=0; }

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

a) [1] Sabendo que o código em *assembly* da função `stay` é o indicado. Qual é o valor da constante `ROOMS`?

b) [1] Devido a um *bug*, o programa acede ao *array* `stays` com a expressão `stays[2][-2]`. Qual o elemento do *array* que é realmente acedido?

c) [2] Apresente uma implementação em IA-32 da função `procStays`.

```
stay:
  push  ebp
  mov   ebp, esp
  mov   eax, [ebp+8]
  lea   eax, [eax+eax*4]
  add   eax, eax
  add   eax, [ebp+12]
  pop   ebp
  mov   eax, [stays+eax*4]
  ret
```

d) [2] Descreva todo o conteúdo do ficheiro `findStay.c` com a implementação da função `findStay` usando a função `procStays`. A função `findStay` retorna um ponteiro para a primeira estadia encontrada do cliente com o nome indicado, ou retorna NULL se não encontrou nenhuma com esse cliente.

e) [2] Implemente em IA-32 a função `setStay` num módulo `setStay.s`. Esta função preenche uma estrutura `Stay` com os elementos indicados (n-`name`, f-`floor` e r-`room`) copiando o nome do cliente para memória alojada dinamicamente através da função `char * strdup(const char*s)`. A função `strdup` retorna um ponteiro para o espaço alojado dinamicamente contendo uma cópia da *string* passada como parâmetro.

f) [1] Implemente em C da função `enter` que pertence ao módulo `Hotel.c`. Esta função usa a função `setStay` para acrescentar uma estadia no *array* `stays` que será mais tarde removida com a função `leave`. Se a estadia para o andar e quarto indicados já existir, ela será apenas actualizada.

g) [1] Faça o *makefile* para gerar o programa completo, compilando separadamente cada um dos módulos.

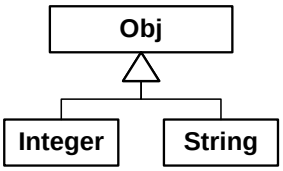
4. [5] Para realizar um programa, desenvolvido na linguagem C, com as técnicas usadas na programação orientada por objectos, considere os ficheiros fonte `Obj.h` e `Objs.c` seguintes e o diagrama que representa a hierarquia de classes pretendida.

```
typedef void (*DelFx)(void* this);
typedef void (*ToStrFx)(void* this,
                        char *str);

typedef struct _vtbl {
  const char *typeName;
  DelFx delete;
  ToStrFx toString;
} ObjVtbl;

typedef struct _obj {
  ObjVtbl *vt;
} Obj;

#define typeOf(o) (o)->vt->typeName
```




```
#include "Obj.h"
#include "Integer.h"
#include "String.h"

void printAll( Obj **a, unsigned dim ) {
  char buffer[512];
  for( ; dim ; --dim, ++a ) {
    (*a)->vt->toString(*a,buffer);
    printf("%s:%s ",typeOf(*a),buffer);
} }

void deleteAll( Obj **a, unsigned dim )
{ for(; dim ; --dim,++a) (*a)->vt->delete(*a); }

Obj * newInstance(const char *t,const char *v) {
  if (!strcmp(t,"Integer")) return (Obj*)newInteger(atoi(v));
  if (!strcmp(t,"String")) return (Obj*)newString(v);
  return NULL;
}

int main(int argc, const char *argv[]) {
  unsigned int len=0, j, dim = (argc-1)/2;
  Obj **objs= malloc(sizeof(Obj*)*dim);
  for( j=1 ; j<argc ; j+=2)
    if ((objs[len]=newInstance(argv[j],argv[j+1]))) ++len;
  printAll(objs,len);
  deleteAll(objs,len); free(objs); return 0;
}
```

O programa seria composto pelos módulos `Objs.o` `Integer.o` e `String.o`. Uma possível linha de comandos para executar o programa poderia ser: `Objs Integer 27 String ISEL Integer 81` e o respectivo output, gerado na função `printAll`, será: `Int:27 Str:ISEL Int:81 .`

a) [3] Assumindo a existência dos ficheiros `String.h` e `String.c` para implementar a entidade `String`, escreva o conteúdo do ficheiro *header* `Integer.h` e o ficheiro fonte `Integer.c` para implementar a "classe" *Integer*.

b) [2] Descreva sumariamente as alterações que teria que realizar para o programa carregar em tempo de execução, a partir de uma biblioteca dinâmica, o código de qualquer "classe" derivada de *Obj*.

Duração: 2 horas e 30 minutos

Bom teste!