1. [7] Considere as seguintes definições em linguagem C. Trata-se da implementação de um *buffer* de dimensão limitada, com disciplina FIFO (*first in, first out*), para armazenamento de linhas de texto. O alojamento das linhas de texto é controlado pelo utilizador do `Fifo`.

```
typedef struct {
   char * * buffer;
   int put, get, count, size;
} Fifo;

Fifo * fifo_create(int size);

void fifo_destroy(Fifo * fifo);

int fifo_full(Fifo * fifo);

int fifo_empty(Fifo * fifo);

void fifo_insert(Fifo * fifo, char * p);
char * fifo_remove(Fifo * fifo);

void fifo_for_each(Fifo * fifo,
   void (*print)(char *));
```

```
Fifo * fifo_create(int size) {
   Fifo * fifo =
      (Fifo*)malloc(sizeof(Fifo) + size * sizeof(char *));
    if (fifo == NULL)
      return NULL;
   fifo->buffer = (char **)((char *)fifo + sizeof(Fifo));
   fifo->put = fifo->get = fifo->count = 0;
   fifo->size = size;
  return fifo;
}
void fifo_insert(Fifo * fifo, char * p) {
   fifo->buffer[fifo->put] = p;
   if (++fifo->put == fifo->size) fifo->put = 0;
   fifo->count++;
}
void fifo_for_each(Fifo * fifo, void (*_do)(char *)) {
   int i;
   for (i = 0; i < fifo->count; ++i)
    _do(fifo->buffer[(fifo->get + i) % fifo->size]);
}
```

a) [1] Considere a programação apresentada, em *assembly*, da função **`fifo_empty`**. Programe-a em linguagem C.

```
   .global  fifo_empty
fifo_empty:
   mov   eax, [esp + 4]
   cmp   dword ptr [eax + 12],0
   mov   eax, 1
   jz L1
   mov   eax, 0
L1:
   ret
```

b) [1] Programe a função **`fifo_destroy`** que elimina uma variável do tipo **`Fifo`**.

c) [1] Programe a função **`fifo_remove`** que remove do *buffer* a linha de texto inserida há mais tempo.

d) [2] Traduza a função **`fifo_for_each`** para linguagem *assembly*. Esta função aplica o processamento **`_do`** a cada elemento do *buffer*. Por exemplo, esta função pode ser usada para imprimir todos as linhas de texto contidas no *buffer*, através da seguinte invocação: **`fifo_for_each(fifo, puts);`**

e) [2] Tirando partido de uma variável deste tipo (**`Fifo`**), realize um programa que imprima as últimas n linhas de um ficheiro de texto. Tanto o ficheiro como o número de linhas, devem ser passados como argumentos na linha de comando.

2. [4] Considere o excerto da tabela de símbolos do executável `app` e os ficheiros fonte `app.c, m.c, dm.c e dm2.c`.

app.c
```
extern int func1();
extern int func2(int);
typedef void (*PF)(int*);

int v1=1;
int x=10;

int main() {
   void * l= dlopen("./dm2.so",RTLD_NOW);
   ((PF)dlsym(l,"func4"))(&v1);
   printf("%d\n", func1() + func2(var1));
   return 0;
}
```

m.c
```
extern int x;
static int v1 = 10;
int func1() { return v1 + x; }
```

dm.c
```
static int fs(int v)
{ return v * 1024; }
int func2(int v) {return fs(v);}
```

dm2.c
```
void func4(int *v) {
  *v *= 2;
}
```

```
> nm -n app
         U func2
08048584 T main
08048600 T func1
0804a020 D v1
0804a024 D x
0804a028 d v1

B-bss   D-data
T-text  U-undefined
```

a) [1] Explique o facto do símbolo `func2` estar marcado como *undefined* na tabela de símbolos de `app`.

b) [1] Escreva um possível `Makefile` que gere a aplicação `app` de acordo com a tabela de símbolos apresentada.

c) [1] Indique e justifique o *output* apresentado na execução de `app`.

d) [1] Actualize os valores finais dos símbolos indefinidos presentes nos ficheiros objecto recolocáveis `app.o` e `m.o`, considerando o valor `0xB8092300` para o símbolo `func2`. Considere também que a instrução `CALL` é codificada com um endereço relativo ao PC e que o acesso a uma variável é codificada com um endereço absoluto, ambos codificados em *little-endian*.

```
> objdump -d -M intel app.o m.o

app.o
00000000 <main>:
5a: e8 __ __ __ __ call  <func1>
67: e8 __ __ __ __ call  <func2>
ca: c3             ret

m.o
00000000 <func1>:
03:  a1 __ __ __ __ mov   eax,<x>
09:  a1 __ __ __ __ mov   eax,<v1>
0c:  c3           ret
```

Indique, justificando, a fase no processo de geração da aplicação onde cada símbolo poderá ser resolvido.
Note que apenas estão apresentadas as linhas de código a necessitar de actualização.
Justifique devidamente todos os cálculos efectuados.

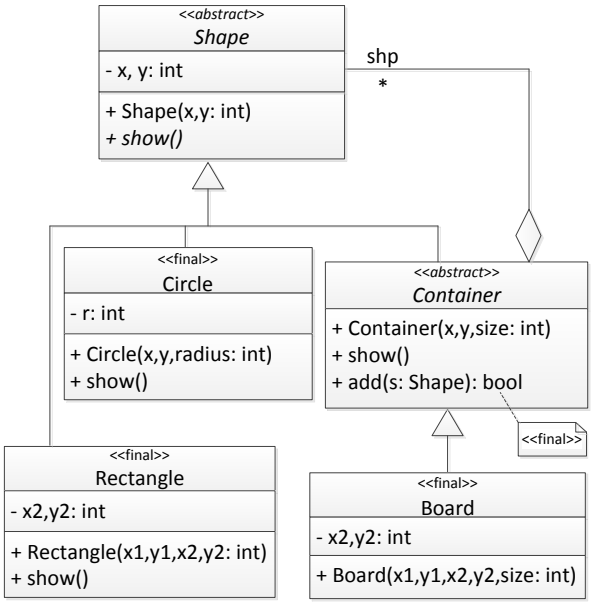
3. [1] Implemente a macro **`SUM(T, V, R)`** que coloca no inteiro **`R`** a soma dos *bytes* do valor **`V`** do tipo primitivo **`T`**. Por exemplo, a execução do código ao lado coloca em r o valor 10.

```
int i = 0x01020304;
int r;
SUM(int, i, r);
```

4. [2] Considere um sistema com arquitectura IA-32, e uma *cache* de dados *4-way set-associative*, sendo 16 bits do endereço usados como *tag*. Sabendo que uma instância da estrutura `Thing` pode ocupar exactamente 2 linhas de cache, qual a dimensão da *cache*? **Nota: na resposta à questão considere que o compilador definiu o *layout* da estrutura de forma a optimizar o acesso aos respectivos campos.**

```
struct Thing {
  int   id;
  char  type;
  int age;
  char  name[20];
};
```

5. [6] Considere o diagrama de classes da figura. A classe `Shape` representa uma forma geométrica (FG) constituída por uma origem `x`, `y` e pelo método abstracto `show` que deverá apresentar na consola a informação da FG. A classe `Container` é uma FG que agrega outras FG's. Na construção é indicada a sua origem e o número máximo de FG's possíveis de agregação. O método `show` apresenta na consola a origem do próprio contentor e a informação de todas as FG's agregadas. O método `add` agrega a FG `s` ao contentor se não foi atingido o número máximo de FG's agregadas e o retorno reflecte o sucesso da operação. Assume-se que as FG's adicionadas a um contentor foram alocadas dinamicamente. A destruição de um contentor implica a libertação da memória ocupada por todas as FG's agregadas.

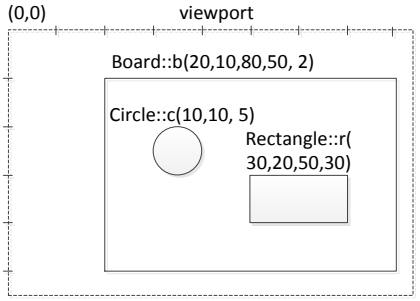


a) [3,5] Escreva na linguagem C uma definição equivalente dos tipos `Shape` e `Container`.

b) [2,5] Escreva um programa que crie em memória um modelo equivalente ao da figura e que apresente na consola o resultado da chamada ao método `show` do `Board b`. No final o programa deverá libertar todos os recursos alocados. Considere a existência das funções:

```
void circle_init(Circle *obj, int x, int y, int r);
void rectangle_init(Rectangle *obj, int x1, int y1, int x2, int y2);
void board_init(Board *obj, int x1, int y1, int x2, int y2, int size);
```

que iniciam respectivamente objectos do tipo `Circle`, `Rectangle` e `Board`.



Duração: 2 horas e 30 minutos

Bom teste!