 <small>ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO</small>	Tipo de Prova Teste modelo	Ano letivo 20XX/20XX	Data XX-XX-XXXX
	Curso Lic. em Engenharia Informática Lic. em Segurança Informática e Redes de Computadores	Hora XX:XX	
	Unidade Curricular Fundamentos de Programação	Duração XXhXX	

#### Observações

- Preencha todo o cabeçalho da(s) folha(s) de teste: nome completo e número do estudante, data de realização da prova de avaliação, nome da unidade curricular e do curso.
- Se quiser desistir deverá escrever na folha de exame "Desisto" e colocar por baixo a sua assinatura.
- Não é permitido o uso de qualquer documentação além da indicada ou fornecida pelo docente.
- Deverá entregar tudo o que lhe foi entregue pelo docente: folhas de teste, folhas de rascunho e enunciado.
- Os estudantes não devem sair da sala de exame sem assinar a folha de presenças.

## Parte 2

Dado o seguinte *header file*

```
#ifndef SALAS_H
#define SALAS_H

#define SALAS_TAM_INICIAL 5 // quantidade inicial de salas a alocar.
#define SALA_NOME_MAX 51    // limite de caracteres para o nome das salas.


typedef struct {
    int capacidade; // capacidade máxima de alunos na sala.
    int ocupacao;   // número de alunos com presença marcada na sala.
    char nome[SALA_NOME_MAX]; // nome da sala (deve ser único).
    int *presencas; // armazena os números dos alunos presentes na sala. Apontador
                  // para as presenças alocadas dinamicamente com tamanho igual à
                  // capacidade da sala.
} Sala;

typedef struct {
    int contador; // quantidade de salas existentes (inicializado com 0).
    int tamanho;  // quantidade de salas alocadas (inicializado com
                  // SALAS_TAM_INICIAL).
    Sala *salas; // apontador para as salas alocadas.
} Salas;

int carregarSalas(Salas *salas, char *nome_fich);
int inserirSala(Salas *salas, char *nome, int capacidade);
Sala* procurarSala(Salas salas, char *nome);
int removerSala(Salas *salas, char *nome);
void listarSalas(Salas salas);
int libertarSalas(Salas *salas);
int marcarPresencas(Salas *salas, char *nome);
void guardarSalas(Salas salas, char *nome_fich);

#endif /* SALAS_H */
```

1. Implemente a função `void guardarSalas(Salas salas, char *nome_fich)` que armazena no ficheiro (`nome_fich`) toda a informação das `salas` e das presenças em `salas`.

 <small>ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO</small>	Tipo de Prova Teste modelo	Ano letivo 20XX/20XX	Data XX-XX-XXXX
	Curso Lic. em Engenharia Informática Lic. em Segurança Informática e Redes de Computadores	Hora XX:XX	
	Unidade Curricular Fundamentos de Programação	Duração XXhXX	

2. Tendo em conta a função anterior, implemente a função **int carregarSalas(Salas \*salas, char \*nome\_fich)** que recupere do ficheiro (**nome\_fich**) toda a informação previamente armazenada das salas e das presenças em **salas**. A função retorna **1** se terminar com sucesso; **0**, caso contrário.
3. Implemente a função **int inserirSala(Salas \*salas, char \*nome, int capacidade)** que insere uma nova sala, considerando o nome da sala (deverá verificar se não existe outra sala com o mesmo nome) e a capacidade desta. Deverá alocar a memória necessária para a marcação das presenças (considerando a capacidade da sala). A função retorna **1** se terminar com sucesso; **0**, caso contrário.
4. Implemente a função **int removerSala(Salas \*salas, char \*nome)** que remove de **salas**, a sala (**nome**) dada como parâmetro na função e liberta a memória desnecessária. A função retorna **1** se terminar com sucesso; **0**, caso contrário.