

U.C. 21077

Linguagens de Programação e-Fólio A – Linguagem OCaml e Java

-- INSTRUÇÕES --

- 1) O e-fólio tem uma cotação de 4 valores.
- 2) Qualquer tentativa de plágio resultará numa nota final de zero valores.
- 3) Este e-fólio deve ser resolvido usando a *linguagem Java e OCaml*.
- 4) Deve ser submetido um ficheiro comprimido (ZIP ou RAR) com o nome e número de estudante contendo:
 - a) Código do programa devidamente comentado;
 - b) Ficheiro readme.txt com a informação necessária para compilar e executar o programa;
 - c) Relatório em formato *.pdf* até 4 páginas descrevendo a solução apresentada e os testes efetuados

E-fólio A

A coordenação do curso de Engenharia Informática pretende um sistema de acompanhamento automático de turmas, capaz de analisar o desempenho e participação dos alunos ao longo do semestre e atribuir um estado de aproveitamento baseado em regras de decisão predefinidas.

A base de dados, fornecida por um antigo professor que usava Prolog, contém factos sobre:

- Alunos inscritos.
- Atividades realizadas (fórum, tarefa, quiz).
- Registos de assiduidade em aulas práticas.
- Autoavaliações submetidas pelos alunos.

A partir destes dados, o sistema deve calcular indicadores individuais e, através de uma árvore de decisão, classificar cada aluno com um estado final. A complexidade da análise exige que todo o processamento seja feito em OCaml, enquanto uma interface em Java apenas invoca o executável OCaml e apresenta os resultados de forma consolidada.

A coordenação entregou-lhe uma base de dados no ficheiro “database.pl”.

Ao falar com vários docentes estes informaram as regras de decisão. As seguintes regras devem ser aplicadas a cada aluno:

Regra	Descrição	Resultado Parcial
R1	O aluno tem pelo menos 3 participações no fórum?	Participação adequada
R2	A média aritmética das notas de tarefas e quizzes (conjunta) é ≥ 10 valores?	Desempenho positivo
R3	A assiduidade (percentagem de aulas assistidas) é $\geq 75\%$?	Assiduidade regular
R4	A nota de autoavaliação está dentro de ± 2 valores da média de tarefas e quizzes?	Autoavaliação coerente

Estados finais (aplicar pela ordem indicada):

Condições satisfeitas	Estado Final
Participação adequada E Desempenho positivo E Assiduidade regular	"Aprovado"
Participação adequada E Desempenho positivo MAS Assiduidade $< 75\%$	"Condicionado"
Apenas Desempenho positivo (independentemente das restantes)	"Em Observação"
Participação adequada OU Desempenho positivo, mas não ambos	"Em Risco"
Nenhuma das anteriores	"Retido"

Caso a autoavaliação seja **coerente**, o estado final é melhorado um nível (por exemplo, "Em Risco" passa a "Em Observação"; "Em Observação" passa a "Condicionado"; "Condicionado" passa a "Aprovado"). "Aprovado" com coerência mantém-se "Aprovado". "Retido" nunca é melhorado.

1. Leitura e Organização da Base de Dados (OCaml) - 0,4 Valores

O arquivo database.pl contém dados em formato Prolog.

Implemente funções para:

- a) Ler a base de dados (*para o ajudar veja o exemplo abaixo de como se lê e recolhe os itens em inventário de provas anteriores ex.: ./main.exe listar_items*)
- b) Listar todos os alunos ordenados alfabeticamente por nome, apresentando os seus dados formatados (./main.exe listar_alunos): ID; Nome; Email; Nº Atividades; Assiduidade (%)

```
let read_file filename =
  let lines = ref [] in
  let channel = open_in filename in
  try
    while true do
      lines := input_line channel :: !lines
    done; []
  with End_of_file ->
    close_in channel;
    List.rev !lines

let parse_item line =
  let regexp = Str.regexp "item(\\([0-9]+\\), '\\([^']+\\)', '\\([^']+\\)', '\\([^']+\\)', '\\([0-9.]+\\)', '\\([0-9.]+\\)', '\\([0-9]+\\))." in
  if Str.string_match regexp line 0 then
    Some (
      int_of_string (Str.matched_group 1 line), (* ID *)
      Str.matched_group 2 line,                 (* Nome *)
      Str.matched_group 3 line,                 (* Marca *)
      Str.matched_group 4 line,                 (* Tipo *)
      float_of_string (Str.matched_group 5 line), (* Custo *)
      float_of_string (Str.matched_group 6 line), (* Preço Venda *)
      int_of_string (Str.matched_group 7 line)  (* Quantidade *)
    )
  else
    None

let () =
  match Array.to_list Sys.argv with
  | _ :: "listar_items" :: _ ->
    let file_lines = read_file "/home/rudigualter/projects/e-folioA/database/database.pl" in
    let itens = List.filter_map parse_item file_lines in
    print_items itens
```

```
| _ ->
Printf.printf "Comandos Disponíveis:\n";
Printf.printf "  listar_items\n";
```

2. Cálculo de Indicadores por Aluno (OCaml) – 0,6 Valores

Para um aluno dado o seu ID, calcule e exiba (./main.exe indicadores 1):

- Número de participações no fórum.
- Média das notas de tarefas.
- Média das notas de quizzes.
- Média conjunta (tarefas + quizzes).
- Percentagem de assiduidade.
- Nota de autoavaliação (se existir).

Saída esperada (formato): ID; Participações Fórum; Média Tarefas; Média Quizzes; Média Conjunta; Assiduidade; Autoavaliação

3. Aplicação da Árvore de Decisão (OCaml) – 1,6 Valores

a) Dado o ID de um aluno, exiba a avaliação passo a passo das regras R1 a R4 e o estado final calculado (./main.exe avaliar 2).

Saída esperada: ID; R1; R2; R3; R4; Estado Final

As regras retornam falso ou verdadeiro por exemplo:

R1	(≥ 3	fórum):	false	(1	participação)
R2	(média	≥ 10):	false	(8.5)	
R3	(assid.	$\geq 75\%$):	true	(80%)	
R4	(autoav.	coerente):	false	(auto=6,	média=8.5)
Estado final: Retido					

b) Processar todos os alunos e gerar uma listagem ordenada por estado final (de "Aprovado" a "Retido") e, dentro do mesmo estado, por nome (./main.exe listar_estados). Formato: ID; Estado; Média Conjunta; Assiduidade

4. Sistema de apresentação (Java) – 1 Valor

Desenvolva uma aplicação Java que:

- Utilize ProcessBuilder para executar o programa OCaml compilado (ex.: ./main.exe).
- Leia a saída padrão do processo OCaml e a exiba formatada na consola Java.
- Apresente um menu interativo que permita ao utilizador escolher as opções correspondentes às questões anteriores.
- Emissão de boletim de um aluno, com escrita em ficheiro com o formato JSON e ecrã que apresente os outputs gerados na questão 2 e questão 3.

Requisitos técnicos:

- Código organizado em classes (ex.: Aplicacao, IntegradorOCaml, Menu).
- Tratamento de erros robusto (ex.: ficheiro não encontrado, argumentos inválidos).
- Utilização de boas práticas de programação orientada a objetos (encapsulamento, nomes significativos).

5. Relatório – 0.4 Valor

BONUS: 0.1 Valores para quem criar uma qualquer opção extra e funcional no Ocaml e respetiva interface no Java, como por exemplo “Listar os alunos com autoavaliação coerente”, pode ser qualquer funcionalidade extra.

Notas:

1. (C3) Todas as escolhas devem ser fundamentadas no relatório.
2. (C1) A forma de implementar os vários módulos propostos.
3. (C2) A forma e lógica do programa fica ao critério de cada estudante, mantendo as boas práticas de programação.
4. (C2) A facilidade de utilização do programa é valorizada (exemplo: menus de acesso, e outras estruturas e termos complexos)

Matriz de Correção:

Item	Peso	% item	Valor item	Tipo de Erro	Penalização (%)	Desconto	Descrição
1.a	0,4	30%	0,12	Leve	5	0,006	Falta de comentário no código, código não legível, falta de boas práticas.
				Médio	20	0,024	Erros na leitura ou impressão da lista, (saltos), falta de tratamento de erros.
				Grave	50	0,06	Chamada função não funciona, leituras não funcionam.
1.b	0,4	70%	0,28	Leve	5	0,014	Falta de comentários no código, código não legível, falta de boas práticas.
				Médio	20	0,056	Erros na leitura ou impressão da lista, (saltos), falta de tratamento de erros.
				Grave	50	0,14	Chamada da função não funciona; leituras não funcionam.
2	0,6	100%	0,60	Leve	5	0,03	Falta de comentário no código, código não legível, falta de boas práticas.
		100%	0,60	Médio	20	0,12	Cálculo das Médias funciona, mas errado, impressão da lista, (saltos) falta de tratamento de erros.
		100%	0,60	Grave	50	0,3	Chamada função não funciona.

3	1,6	100%	1,60	Leve	5	0,08	Falta de comentários no código, código não legível, falta de boas práticas.
				Médio	20	0,32	Lógica de regra com erros. falta de tratamento de erros.
				Grave	25	0,4	Penalização por falta de cada regra.
				Gravíssimo	50	0,8	Chamada da função não funciona, Regras não funcionam.
4	1	100%	1,00	Leve	5	0,05	Falta de comentários no código, código não legível, falta de boas práticas.
				Médio	20	0,2	Leitura de output com falhas, falta de tratamento de erros, falta de funcionalidades.
				Grave	50	0,5	Sem organização de classes.
5	0,4	100%	0,40	Leve	5	0,02	Falta de organização do documento.
				Médio	20	0,08	Falta de overview do trabalho, listagem de opções tomadas e desafios.
				Grave	50	0,2	Sem provas de teste, sem justificações.