P.PORTO

ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Tipo de Prova	Ano letivo	Data
Teste/exame (EN)	2021/2022	11-02-2022
Curso: Lic. em Engenharia Informática; Lic. em Segurança Informática e Redes de Computadores		Hora: 10:00
Unidade Curricular: Fundamentos de Programação		Duração: 02h00

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.

As mensagens com discursos de ódio postadas nas redes sociais têm o intuito de denegrir uma pessoa ou grupo de pessoas por pertencerem a um dado grupo com base na raça, minorias étnicas, orientação sexual, identidade de género, religião, filiação partidária, deficiência ou opinião. Uma aplicação de Inteligência Artificial (AI App) baseada em técnicas avançadas de Aprendizagem Automática (*Machine Learning*) permite detetar indícios dessas mensagens (*tweets*) na rede social atualmente líder na transmissão de opiniões: o Twitter.
No sentido de controlar (e tentar evitar) este tipo de linguagem, pretende-se monitorizar os *tweets* filtrados e classificados pela AI App com os tipos de ódio envolvidos. Para modelar o sistema proposto, esta AI App devolve o ficheiro **tweets_dados_odio.bin** que contém os dados recolhidos pela AI App com a estrutura de dados apresentada no *header file* **TWEETS.h**. O identificador de cada *tweet* é um número inteiro longo que a rede social Twitter devolve no formato string. Não há limite no número de *tweets*. Os tipos de ódio por *tweet*, classificados pela AI App, estão limitados a um máximo de 8 e numerados da seguinte forma: 1-Racismo, 2-Minorias étnicas, 3-Sexismo, 4-Identidade de Género, 5-Religião, 6-Partido, 7-Deficiência e 8-Opinião.

```
#ifndef TWEETS_H
#define TWEETS_H

#define MAX_ID_STR 20      // limite de caracteres (numéricos) para o identificador.
#define MAX_TEXTO_STR 280  // limite máximo de caracteres para o texto no Twitter.
#define MAX_EMAIL_STR 50   // limite de caracteres para os emails.
#define MAX_TIPOS_ODIO 8   // Número máximo de tipos de ódio

typedef struct {
    unsigned int dia,mes,ano,hora,min,seg; // dia, mês, ano, hora, minuto e segundo.
} DataHora;

typedef struct {
    char id[MAX_ID_STR]; // identificador único do tweet.
    char email[MAX_STR]; // e-mail do autor do tweet.
    char texto[MAX_TEXTO_STR]; // mensagem de texto do tweet.
    DataHora datahora; // data e hora em que o tweet foi criado.
    int n_tipos; // quantidade de tipos de ódio classificados pela AI App do tweet.
    int *tipos; // apontador para os tipos de ódio (máximo 8).
} Tweet;

typedef struct {
    int n_tweets; // quantidade de tweets existentes (ver questão 1).
    Tweet *tweets; // apontador para os tweets alocados.
} Tweetlist;
```

P.PORTO — ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO	Tipo de Prova Teste/exame (E/N)	Ano letivo 2021/2022	Data 11-02-2022
	Curso Lic. em Engenharia Informática Lic. em Segurança Informática e Redes de Computadores		Hora 10:00
	Unidade Curricular Fundamentos de Programação		Duração 02h00

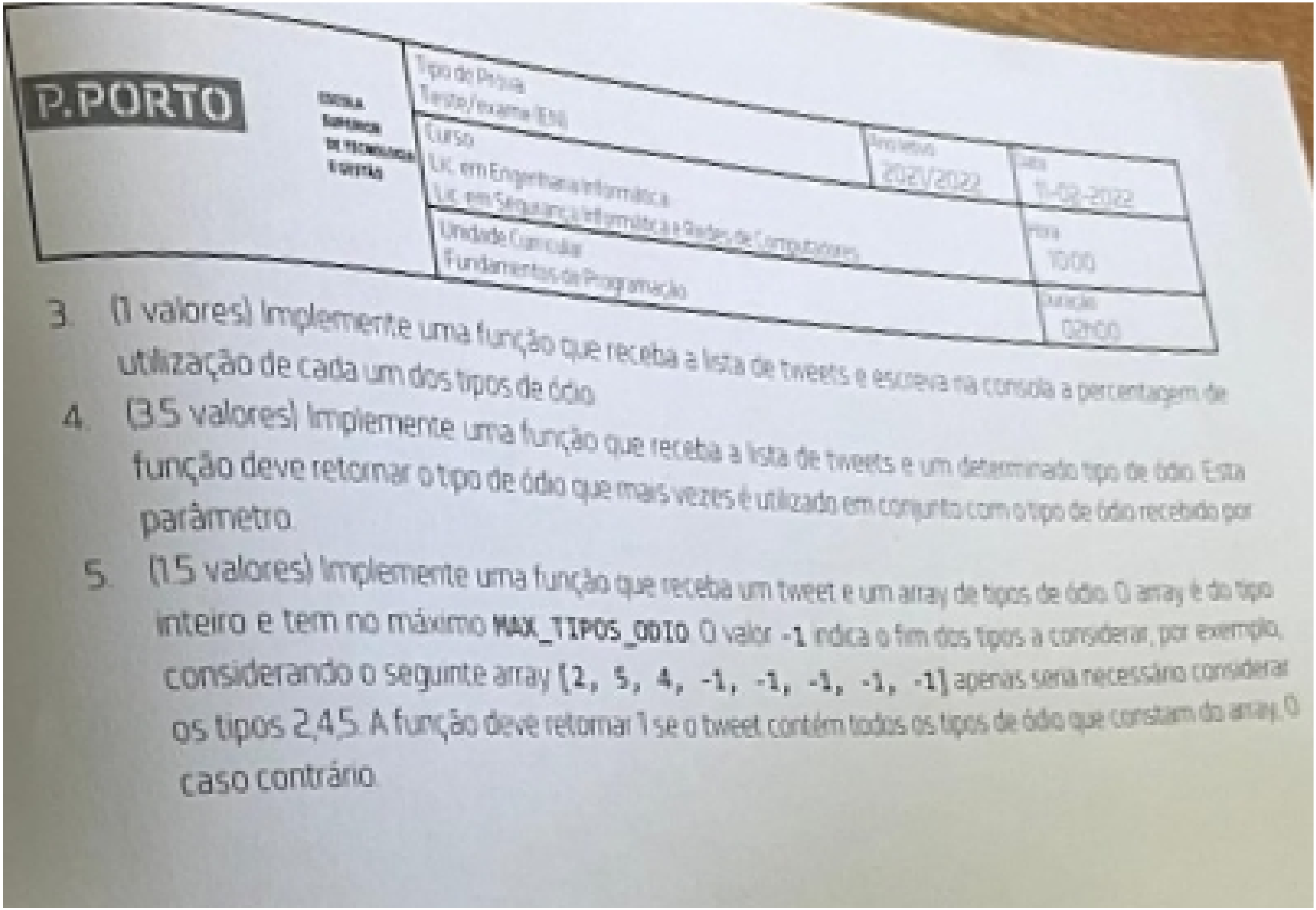
```
int carregarTweetlist(Tweetlist *tl, char *nome_fich);
Tweet* obterTweet(Tweetlist tl, char *id);
char* maxOdiosTweetlist(Tweetlist tl);
void resumoOcorrTiposOdio(Tweetlist tl, int *ocorr);
void libertarTweetlist(Tweetlist *tl);

#endif /* TWEETS_H */
```

Parte 1

1. (2 valores) Implemente a função **int carregarTweetlist(Tweetlist *tl, char *nomeficheiro);** que dado um apontador para um registo do tipo **Tweetlist** e um apontador para o ficheiro de dados devolvido pela AI App, permita carregar todos os dados da lista de tweets contidos num ficheiro binário denominado **tweets_dados_odio.bin** para uma estrutura de memória alocada dinamicamente. O primeiro valor gravado no ficheiro indica a quantidade de tweets existentes (**n_tweets**) no ficheiro. A função retorna 1 se terminar com sucesso, 0, caso contrário. Esta função destina-se a ser invocada no início do programa.
2. (1.5 valores) Implemente a função **Tweet* obterTweet(Tweetlist tl, char *id);** que dada uma variável do tipo **Tweetlist** e um apontador para o identificador do tweet, devolva um apontador para o registo **Tweet**, caso exista um tweet com esse identificador. A função retorna um apontador para o registo **Tweet**, se o tweet existe; **NULL**, caso contrário.
3. (2.5 valores) Implemente a função **char* maxOdiosTweetlist(Tweetlist tl);** que dada uma variável do tipo **Tweetlist**, devolva um apontador para a mensagem de texto que referencie o maior número distinto de tipos de ódio. Existindo mais do que um tweet com o mesmo número máximo de ódios, a função retorna um apontador para a primeira mensagem de texto encontrada; retorna **NULL**, caso a lista de tweets se encontre vazia.
4. (2.5 valores) Implemente a função **void resumoOcorrTiposOdio(Tweetlist tl, int *ocorr);** que dada uma variável do tipo **Tweetlist**, preencha o array **ocorr** com um resumo do número total de ocorrências por tipo de ódio existente na lista de tweets.
5. (1.5 valores) Implemente a função **void libertarTweetlist(Tweetlist *tl);** que dado um apontador para um registo do tipo **Tweetlist**, liberte toda a alocação de memória efetuada. Esta função destina-se a ser invocada no final do programa.

Parte 2

1. (2 valores) Considerando a existência do *header file* **TWEETS.h** implemente um programa que disponibilize um menu (que deve ser disponibilizado até que o utilizador escolha a opção **0**) para as funções **obterTweet**, **maxOdiosTweetlist** e **resumoOcorrTiposOdio**. Deve considerar o carregamento do ficheiro e alocação de memória (**carregarTweetlist**) e a libertação da memória (**libertarTweetlist**).
2. (2 valores) Implemente uma função que receba um **tweet** e verifique se o email é válido. É considerado um email válido se:
   a. Tem pelo menos 6 caracteres.
   b. Contem um @, com pelo menos um caracter antes (x@x.xx é válido, @x.xx, é inválido).
   c. A seguir ao @ (referido na alínea anterior) deve existir um . (ponto). Entre o @ e o ponto deve existir pelo menos um caracter.
   d. A seguir ao . (ponto) (referido na alínea anterior) devem existir pelo menos 2 caracteres.



3. (1 valores) Implemente uma função que receba a lista de tweets e escreva na consola a percentagem de utilização de cada um dos tipos de ódio.
4. (3.5 valores) Implemente uma função que receba a lista de tweets e um determinado tipo de ódio. Esta função deve retornar o tipo de ódio que mais vezes é utilizado em conjunto com o tipo de ódio recebido por parâmetro.
5. (1.5 valores) Implemente uma função que receba um tweet e um array de tipos de ódio. O array é do tipo inteiro e tem no máximo `MAX_TIPOS_ODIO`. O valor `-1` indica o fim dos tipos a considerar, por exemplo, considerando o seguinte array `{2, 5, 4, -1, -1, -1, -1, -1}` apenas seria necessário considerar os tipos 2,4,5. A função deve retornar 1 se o tweet contém todos os tipos de ódio que constam do array. 0 caso contrário.