CEN 230 - System Programming

Spring 2023 - Midterm Exam

12/04/2023

1 (20 pts.) Memory map

Draw the memory map of the code segment given below.
Use `unk` or `?` to indicate unknown memory contents.

```
int a[3] = {1, 0, 0};
int *p = &(a[1]);
int *q = p++;
*p = *q;
a[2] = p;
p = (int*) malloc(3*sizeof(int));
```

2 (20 pts.) Dynamically allocated memory

Find the error(s) in the code segment below. Briefly explain how each error may be corrected (if possible).
Line numbers are added for your reference.

```
/*Ln 1*/ double d = 3;
/*Ln 2*/ double** ptrRef;
/*Ln 3*/ double* ptr = (double*) malloc(17);
/*Ln 4*/ ptr++;
/*Ln 5*/ ptrRef = &ptr;
/*Ln 6*/ ptr = &d;
```

3 (20 pts.) String processing

Write a function that takes a string (i.e., a `char` array) as input and returns the number of words within it. Assume that words are delimited by the blank space character `' '`.
Assume that the input array ends with the EOS (end-of-string) character with value 0. Code against multiple spaces delimiting the same two words. Code against a completely empty string, in which case your code should return 0.
Do not use the string library `string.h` in your answer.
Examples: word count for `"hello world"` is 2, same for `"  hello world"` is still 2, same for `"  "` is 0.
The signature of your function will be as shown below.

```
int countNumWords(char* arr)
```

4 (20 pts.) Command line arguments

Write a complete program that implements a command line currency converter from US Dollars (USD) to Turkish Lira (TRY). Your program will take 2 arguments (in addition to program name): (a) currency rate for USD/TRY - a float, and (b) USD amount to be converted - a float. Code against insufficient/improper/malformed arguments.
You may use function `atof` from `stdlib.h`.
Console output for sample runs:

```
>./convertUSD2TRY.exe 19.25 2.00
38.50
>./convertUSD2TRY.exe 19.25
Insufficient or incorrect arguments, enter 2 floats
>./convertUSD2TRY.exe hello
Insufficient or incorrect arguments, enter 2 floats
```

5 (20 pts.) Dissimilarity matrix

A distance function measures how far away a pair of items are. A simple example is the squared distance on `int` variables. Let x and y be two `int`s. Squared distance between x and y is computed as: $d(x, y) = (x - y)^2$. Here are some examples: $d(4, 7) = 9$, $d(5, -2) = 49$. Notice that squared distance is symmetric: $d(x, y) = d(y, x)$ for any x and y.
Given a collection of n items, a dissimilarity matrix contains all pairwise distances within the collection. When the underlying distance function is symmetric (as in the case of squared distance), dissimilarity matrices are built in a triangular form, containing only the lower or upper half of an $n \times n$ rectangular matrix).
Below is an sample dissimilarity matrix M built on the array `int arr[] = {4, 7, 5, -2};` for the lower half, using squared distance.

0			
9	0		
1	4	0	
36	81	49	0

Table 1: Sample dissimilarity matrix M over array arr

Notice the following about `M`: it is 2-dimensional, has 4 rows (i.e., size of arr), row `M[i]` has $(i + 1)$ columns, the diagonal (i.e., `M[i][i]`) is all 0s, since $d(x, x) = 0$. We observe that cell `M[1][0] = d(arr[1],arr[0]) = d(7,4)`$= (7 - 4)^2 = 3^2 = 9$. Cell `M[0][1]` does not exist, because it would have contained the same value.
Write a function that takes an integer array as input, dynamically creates and fills-in a lower-half dissimilarity matrix using squared distance and returns this matrix.
The signature of your function will be as shown below.

```
int** getDissimlarityMatrix(int* arr, int size)
```