Pair Programming 2 Turn In

Name: \_\_\_Braden Bell\_\_\_\_\_\_\_\_\_ Username: c1020a01 \_\_\_\_\_

Partner name: \_Thomas Snyder\_\_\_\_\_\_\_\_ Partner username: c1020a10\_\_\_\_\_\_\_\_\_

\_**X**\_ I certify that my partner worked with me on this assignment.

SCORE: \_\_\_\_\_\_\_\_\_\_\_\_ (to be filled in by instructor)

2a (5 points)

Text

Description automatically generated

**Main.c:**

/\* File: exampleStart.c

\* Author: Cindy Arnold

\* Description: read/write names and salaries from/to a file

\*/

#include "example.h"

#define MAX\_NAME\_LENGTH 21

#define ARRAY\_LENGTH 5

int main()

{

Employee e[ARRAY\_LENGTH];

int i;

int count;

count = readData( e );

printData( e, count );

return( 0 );

}

**Example.h:**

/\* File: exampleStart.c

\* Author: Cindy Arnold

\* Description: read/write names and salaries from/to a file

\*/

#ifndef EXAMPLE\_H

#define EXAMPLE\_H

#define MAX\_NAME\_LENGTH 21

#define ARRAY\_LENGTH 5

typedef struct employee {

char name[MAX\_NAME\_LENGTH];

double salary;

} Employee;

int readData( Employee[] );

void printData( Employee[], int );

#endif

**Example.c:**

/\* File: exampleStart.c

\* Author: Cindy Arnold

\* Description: read/write names and salaries from/to a file

\*/

#include <stdio.h>

#include "example.h"

#define MAX\_NAME\_LENGTH 21

#define ARRAY\_LENGTH 5

/\* readData: reads data into array of Employee structures from stdin

\* Parameters

\* a the array

\* Returns: number of elements read in

\*/

int readData( Employee e[] ) {

int count = 0;

while( count < ARRAY\_LENGTH &&

scanf( "%s%lf", e[count].name, &e[count].salary ) != EOF ) {

count++;

}

return count;

}

/\* printData prints array of Employee structures to stdout in a formatted

\* table

\* Parameters

\* e array of Employee structures

\* n number of valid elements in array

\* Returns nothing

\*/

void printData( Employee e[], int n ) {

int i;

// print table heading

printf("Name Salary\n");

printf("---------------------------------\n");

for( i = 0; i < n; i++ ) {

printf("%-20s $%10.2lf\n", e[i].name, e[i].salary );

}

}

**Makefile:**

all: myprogram

myprogram: main.o example.o

gcc main.c example.c -o exampleExe

main: main.c example.h

gcc -c main.c

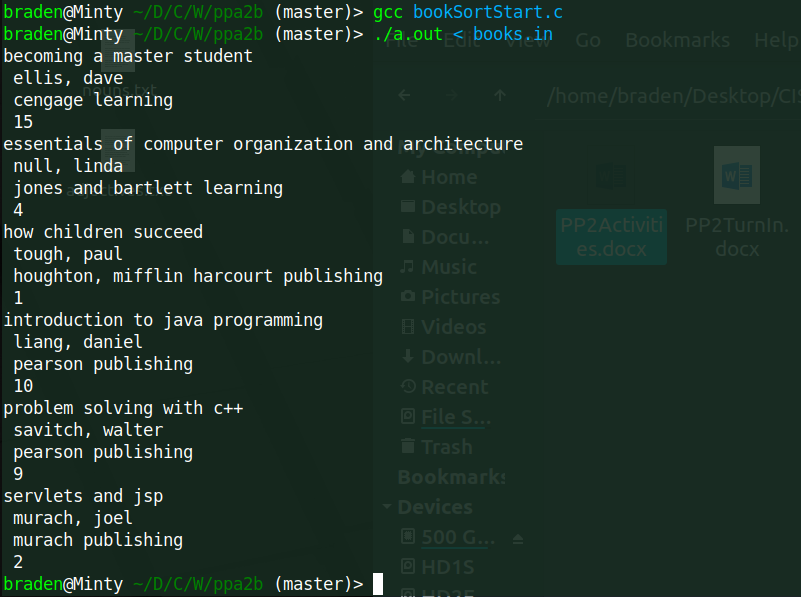
example: example.c example.h

gcc -c example.c

clean:

rm -rf \*.o \*.gch

2b (5 points)



/\* File: bookSort.c

\* Author: Cindy

\* Description: uses array of pointers to an

\* array of structures to sort

\*/

#include <stdio.h>

#include <string.h>

#define MAX\_CHARS 81

#define MAX\_BOOKS 10

typedef struct {

char title[MAX\_CHARS];

char author[MAX\_CHARS];

char publisher[MAX\_CHARS];

int edition;

} book\_t;

int readBooks( book\_t\* );

void initSortingArray( book\_t\*, book\_t\*\*, int );

void printBooks( book\_t\*\*, int );

void sortBooks( book\_t\*\*, int );

void swap( book\_t\*\*, book\_t\*\* );

int main() {

book\_t books[MAX\_BOOKS];

book\_t\* titlePtrs[MAX\_BOOKS];

int numBooks;

numBooks = readBooks( books );

initSortingArray( books, titlePtrs, numBooks );

sortBooks( titlePtrs, numBooks );

printBooks( titlePtrs, numBooks );

return 0;

}

/\* readBooks: fills book\_t array with at most MAX\_BOOKS

\* book info from file books.in

\* Parameters: a array of book structure

\* Returns: number of books

\*/

int readBooks( book\_t\* a ) {

FILE\* in;

int i = 0;

char junk;

in = fopen( "books.in", "r" );

if ( in != NULL ) {

while ( i < MAX\_BOOKS &&

fgets( a[i].title, MAX\_CHARS, in ) != NULL ) {

fgets( a[i].author, MAX\_CHARS, in );

fgets( a[i].publisher, MAX\_CHARS, in );

fscanf(in, "%d", &a[i].edition );

fscanf(in, "%c", &junk );

i++;

}

}

return i;

}

/\* initSortingArray: sets all pointers in array to point to

\* books array

\* Parameters:

\* a books array

\* t pointers to elements in books array

\* n number of books in array a

\* Postcondition: t contains n valid pointers to n elements in a

\* Returns: nothing

\*/

// ADD CODE HERE for the initSortingArray that initializes the array of pointers

void initSortingArray(book\_t\* a, book\_t\*\* t, int n ) {

for (int i = 0; i < n; i++) {

t[i] = &a[i];

}

}

/\* printBooks: prints book info to stdout

\* Parameters:

\* aPtr: array of pointers to book\_t structures

\* n: number of valid element in n

\* Returns: nothing

\*/

void printBooks( book\_t\*\* aPtr, int n ) {

int i;

for( i = 0; i < n; i++ ) {

printf("%s %s %s %d\n", aPtr[i]->title, aPtr[i]->author,

aPtr[i]->publisher, aPtr[i]->edition );

}

}

/\* sortBooks: uses a selection sort to sort book info

\* Parameters:

\* aPtr: array of pointers to book\_t structures

\* n: number of valid elements in aPtr

\* Returns: nothing

\*/

// ADD CODE HERE for the sort function that sorts/swaps the array of pointers

// NOT the array of books by book title

// MAKE SURE TO CALL THE swap FUNCTION!

void sortBooks(book\_t\*\* bookPtrs, int n) {

int i = 0;

int minI = 0;

int top = 0;

for (top = 0; top < (n - 1); top++) {

minI = top;

for (int i = (top + 1); i < n; i++) {

if(strcmp(bookPtrs[i]->title, bookPtrs[minI]->title) < 0) {

minI = i;

}

}

swap(&bookPtrs[minI], &bookPtrs[top]);

}

}

/\* swap: swaps pointers to book\_t array elements

\* Parameters:

\* b1, b2: references to pointers to book\_t elements

\* Postcondition: actual parameter pointer values swapped

\* Returns: nothing

\*/

// ADD CODE HERE TO swap the pointers to books in the array of pointers

void swap(book\_t\*\* b1 , book\_t\*\* b2) {

book\_t\* temp = \*b1;

\*b1 = \*b2;

\*b2 = temp;

}