9.1.a

The difference between calling a function by value and by reference.

The function may be passed copy of the arguments or references to the actual

arguments themselves.

While the function is running if it has access to the reference then it can

change it on the fly from inside the function but if the arguemnt is copied then

you need to use the return value to modify the original.

9.1.b

Limitations of arrays. At least three.

Arrays are sometimes more cumbersome and less efficient than they need to be.

Arrays have a set size.

Arrays hold only one data type.

9.1.c

strcmp(char \*, char \*) function.

The string comparison function allows you to compare strings to see if they are

equal to, greater than or less than each other.

The function will return 0 if they are the same.

Question 2

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main(void) {

/\* Constants \*/

/\* Variables \*/

struct Student {

int stuID;

char name[30];

int mark[3];

};

/\* I can't be bothered typing struct so I'm using typedef. \*/

typedef struct Student Student;

/\* Make some fake variables. \*/

Student stu0001;

stu0001.stuID = 0001;

strcpy( stu0001.name, "Boris Johnson" );

stu0001.mark[0] = 100;

stu0001.mark[1] = 50;

stu0001.mark[2] = 0;

/\* This function calculates the average value of all of the values in an

\* array you pass to it.

\* I figured that an int would be accurate enough, no floats here.

\* I actually wrote this function before reading task 9.3 so I'm leaving it

\* here. The next iteration will take the whole student struct then find the

\* mark value and average that instead of passing just the stux->mark. \*/

int getAverageFromArray( int\* a ) {

int size = sizeof( a ) / sizeof( \*a ) + 1;

int result = 0;

int i = 0;

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

result = result + a[i];

}

return result / size;

}

/\* !!! CURRENTLY DOES NOTHING !!! \*/

/\* Ask which student to check details of. \*/

printf( "Which student would you like to view the details of?\n" );

char input[10];

fflush(stdin);

fgets(input, 10, stdin);

printf( "%s", input );

/\* This is just an example we don't care what the user input. \*/

printf( "Did you mean Boris Johnson?\n\n" );

/\* Make a pointer and grab the student info that the user wanted. \*/

Student\* stux;

stux = &stu0001;

/\* Work out the student's average mark. \*/

int stuxAverage;

stuxAverage = getAverageFromArray( stux->mark );

/\* I tried to print all of these at once but the comptuer didn't like it for

\* some reason so they're separate now. \*/

printf( "Student Details\n" );

printf( "Student ID: %d\n", stux->stuID );

printf( "Student Name: %s\n", stux->name );

printf( "Assignment 1: %d\n", stux->mark[0] );

printf( "Assignment 2: %d\n", stux->mark[1] );

printf( "Assignment 3: %d\n", stux->mark[2] );

printf( "Average Mark: %d\n", stuxAverage );

return 0;

}

Question 3

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main(void) {

/\* Constants \*/

/\* Variables \*/

struct Student {

int stuID;

char name[30];

int mark[3];

};

/\* I can't be bothered typing struct so I'm using typedef. \*/

typedef struct Student Student;

/\* Make some fake variables. \*/

Student stu0001;

stu0001.stuID = 0001;

strcpy( stu0001.name, "Boris Johnson" );

stu0001.mark[0] = 100;

stu0001.mark[1] = 50;

stu0001.mark[2] = 0;

/\* This function calculates the average value of all of the values in an

\* array you pass to it.

\* I figured that an int would be accurate enough, no floats here. \*/

/\* !!! MAKE THIS TAKE A STRUCT INSTEAD OF THE ARRAY !!! \*/

int getAverageFromArray( Student\* a ) {

int size = sizeof( a->mark ) / sizeof( \*a->mark );

int result = 0;

int i = 0;

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

result = result + a->mark[i];

}

return result / size;

}

/\* !!! CURRENTLY DOES NOTHING !!! \*/

/\* Ask which student to check details of. \*/

printf( "Which student would you like to view the details of?\n" );

char input[10];

fflush(stdin);

fgets(input, 10, stdin);

printf( "%s", input );

/\* This is just an example we don't care what the user input. \*/

printf( "Did you mean Boris Johnson?\n\n" );

/\* Make a pointer and grab the student info that the user wanted. \*/

Student\* stux;

stux = &stu0001;

/\* Work out the student's average mark. \*/

int stuxAverage;

stuxAverage = getAverageFromArray( stux );

/\* I tried to print all of these at once but the comptuer didn't like it for

\* some reason so they're separate now. \*/

printf( "Student Details\n" );

printf( "Student ID: %d\n", stux->stuID );

printf( "Student Name: %s\n", stux->name );

printf( "Assignment 1: %d\n", stux->mark[0] );

printf( "Assignment 2: %d\n", stux->mark[1] );

printf( "Assignment 3: %d\n", stux->mark[2] );

printf( "Average Mark: %d\n", stuxAverage );

return 0;

}

Question 4

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main(void) {

/\* Constants \*/

/\* Variables \*/

struct Student {

int stuID;

char name[30];

int mark[3];

};

/\* I can't be bothered typing struct so I'm using typedef. \*/

typedef struct Student Student;

/\* Make some fake variables. \*/

/\*

Student stu0001;

stu0001.stuID = 0001;

strcpy( stu0001.name, "Boris Johnson" );

stu0001.mark[0] = 100;

stu0001.mark[1] = 50;

stu0001.mark[2] = 0;

\*/

Student students[3];

students[0].stuID = 1;

strcpy( students[0].name, "Boris Johnson" );

students[0].mark[0] = 100;

students[0].mark[1] = 100;

students[0].mark[2] = 19;

students[1].stuID = 2;

strcpy( students[1].name, "Theresa May" );

students[1].mark[0] = 100;

students[1].mark[1] = 19;

students[1].mark[2] = 16;

students[2].stuID = 3;

strcpy( students[2].name, "David Cameron" );

students[2].mark[0] = 100;

students[2].mark[1] = 16;

students[2].mark[2] = 10;

/\* This function calculates the average value of all of the values in an

\* array you pass to it.

\* I figured that an int would be accurate enough, no floats here. \*/

/\* !!! MAKE THIS TAKE A STRUCT INSTEAD OF THE ARRAY !!! \*/

int getAverageFromArray( Student\* a ) {

int size = sizeof( a->mark ) / sizeof( \*a->mark );

int result = 0;

int i = 0;

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

result = result + a->mark[i];

}

return result / size;

}

/\* !!! CURRENTLY DOES NOTHING !!! \*/

/\* Ask which student to check details of. \*/

/\*

printf( "Which student would you like to view the details of?\n" );

char input[10];

fflush(stdin);

fgets(input, 10, stdin);

printf( "%s", input );

\*/

/\* This is just an example we don't care what the user input. \*/

/\*

printf( "Did you mean Boris Johnson?\n\n" );

\*/

/\* Make a pointer to the student info. \*/

Student\* stux;

/\* Iterate through the students using a pointer to the current one we're

\* looking at so I don't have to write [i] 20 times. \*/

int i = 0;

printf( "Student Details\n\n" );

for( i = 0; i < sizeof( students ) / sizeof( \*students ); i++ ) {

stux = &students[i];

/\* Work out the student's average mark. \*/

int stuxAverage;

stuxAverage = getAverageFromArray( stux );

/\* I tried to print all of these at once but the comptuer didn't like it for

\* some reason so they're separate now. \*/

printf( "Student ID: %d\n", stux->stuID );

printf( "Student Name: %s\n", stux->name );

printf( "Assignment 1: %d\n", stux->mark[0] );

printf( "Assignment 2: %d\n", stux->mark[1] );

printf( "Assignment 3: %d\n", stux->mark[2] );

printf( "Average Mark: %d\n", stuxAverage );

printf( "\n" );

}

return 0;

}

Question 5

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <stdbool.h>

#include <signal.h>

int main(void) {

/\* Constants \*/

/\* Variables \*/

struct Student {

int stuID;

char name[30];

int mark[3];

};

/\* I'm lazy. \*/

int i = 0;

/\* We'll need these for user inputs. \*/

char input[30] = {0};

int choice;

bool fail = false;

/\* I can't be bothered typing struct so I'm using typedef. \*/

typedef struct Student Student;

/\* Make a pointer to the student info for use while iterating through

\* arrays. \*/

Student\* stux;

/\* Make some fake variables. \*/

/\*

Student stu0001;

stu0001.stuID = 0001;

strcpy( stu0001.name, "Boris Johnson" );

stu0001.mark[0] = 100;

stu0001.mark[1] = 50;

stu0001.mark[2] = 0;

\*/

Student students[3];

students[0].stuID = 1;

strcpy( students[0].name, "Boris Johnson" );

students[0].mark[0] = 64;

students[0].mark[1] = 100;

students[0].mark[2] = 19;

students[1].stuID = 2;

strcpy( students[1].name, "Theresa May" );

students[1].mark[0] = 56;

students[1].mark[1] = 19;

students[1].mark[2] = 16;

students[2].stuID = 3;

strcpy( students[2].name, "David Cameron" );

students[2].mark[0] = 66;

students[2].mark[1] = 16;

students[2].mark[2] = 10;

/\* This function calculates the average value of all of the values in an

\* array you pass to it.

\* I figured that an int would be accurate enough, no floats here. \*/

/\* !!! MAKE THIS TAKE A STRUCT INSTEAD OF THE ARRAY !!! \*/

int getAverageFromArray( Student\* a ) {

int size = sizeof( a->mark ) / sizeof( \*a->mark );

int result = 0;

int i = 0;

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

result = result + a->mark[i];

}

return result / size;

}

/\* !!! CURRENTLY DOES NOTHING !!! \*/

/\* Ask which student to check details of. \*/

/\*

printf( "Which student would you like to view the details of?\n" );

char input[10];

fflush(stdin);

fgets(input, 10, stdin);

printf( "%s", input );

\*/

/\* This is just an example we don't care what the user input. \*/

/\*

printf( "Did you mean Boris Johnson?\n\n" );

\*/

void printStudDetails( Student\* a ) {

/\* Work out the student's average mark. \*/

int aAverage;

aAverage = getAverageFromArray( a );

/\* I tried to print all of these at once but the comptuer didn't like it for

\* some reason so they're separate now. \*/

printf( "Student ID: %d\n", a->stuID );

printf( "Student Name: %s\n", a->name );

printf( "Assignment 1: %d\n", a->mark[0] );

printf( "Assignment 2: %d\n", a->mark[1] );

printf( "Assignment 3: %d\n", a->mark[2] );

printf( "Average Mark: %d\n", aAverage );

printf( "\n" );

}

void printAllDetails( void ) {

/\* Iterate through the students using a pointer to the current one we're

\* looking at so I don't have to write [i] 20 times. \*/

printf( "Student Details\n\n" );

for( i = 0; i < sizeof( students ) / sizeof( \*students ); i++ ) {

stux = &students[i];

printStudDetails( stux );

}

}

Student\* searchName( char\* studentname ) {

i = 0;

for( i = 0; i < sizeof( students ) / sizeof( \*students ); i++ ) {

stux = &students[i];

if( strstr( students[i].name, studentname ) != NULL ) {

return &students[i];

} else {

/\* Throw an error the lazy incorrect way. \*/

return 0;

}

}

}

Student\* searchID( int studentid ) {

i = 0;

for( i = 0; i < sizeof( students ) / sizeof( \*students ); i++ ) {

stux = &students[i];

if( stux->stuID == studentid ) {

return &students[i];

} else {

/\* Throw an error the lazy incorrect way. \*/

return 0;

}

}

}

void search( int option ) {

/\* Debugging interrupt. \*/

/\*

raise( SIGINT );

\*/

/\* Take input. \*/

printf( "Input search parameter: \n" );

fflush( stdin );

fgets( input, 10, stdin );

/\* Get rid of trailing newline that stops it ever finding anything. \*/

input[strlen( input ) - 1] = '\0';

/\* Search type. \*/

if( option == 0 ) {

printf( "Searching for %s...\n", input );

if( stux = searchName( input ) ) {

printf( "Student found!\n" );

} else {

printStudDetails( stux );

}

} else {

choice = atoi( input );

printf( "Searching for %d...\n", choice );

if( stux = searchID( choice ) ) {

printf( "Student found!\n" );

} else {

printStudDetails( stux );

}

}

}

bool loop = true;

while(loop == true) {

printf("STUDENT LOOKUP\n\n"

"1 - Print all student details\n"

"Search student by:\n"

"2 - Name\n"

"3 - ID\n\n"

"4 - Exit\n\n" );

fflush( stdin );

fgets( input, 10, stdin );

choice = atoi( input );

switch(choice) {

case 1: printAllDetails();

break;

/\* Search by Name. \*/

case 2: printf( "Searching by name...\n" );

search( 0 );

break;

/\* Search by ID. \*/

case 3: printf( "Searching by ID...\n" );

search( 1 );

break;

default:loop = false;

}

}

return 0;

}

