**Problem 1 follows…**  
**Description:** Problem 1 is designed to demonstrate the typedef statement. It consists of a function which receives two instances of the struct using the typedef, from which a difference between two times is returned.

**Program Code (main.c):**  
#include <stdio.h>

#include <stdlib.h>

//timeStruct is a struct type that has 2 fields

typedef struct timeStruct {

int hours;

int minutes;

} timeStructType;

//function to get difference of time in minutes

int TimeDifference(timeStructType theStart, timeStructType theEnd) {

//get theStart and theEnd time length in minutes

int theStartInMins = (theStart.hours \* 60 + theStart.minutes);

int theEndInMins = (theEnd.hours \* 60 + theEnd.minutes);

//initialize timeDifference variable to be returned

int timeDifference;

//select which version of timeDifference to use

if (theEndInMins < theStartInMins) {

timeDifference = (theEndInMins - theStartInMins + 1440);

} else {

timeDifference = (theEndInMins - theStartInMins);

}

//return timeDifference in minutes

return timeDifference;

}

int main()

{

//make instances of timeStruct using the typedef

timeStructType theStart;

timeStructType theEnd;

//fill its fields

theStart.hours = 11;

theStart.minutes = 52;

theEnd.hours = 14;

theEnd.minutes = 43;

//print contents of theStart and theEnd

printf("theStart is HH:%d MM:%d\n", theStart.hours, theStart.minutes);

printf("theEnd is HH:%d MM:%d\n\n", theEnd.hours, theEnd.minutes);

//call TimeDifference (theStart, theEnd). On return, print out the number of minutes

printf("Time from theStart to theEnd: %d minutes\n", TimeDifference(theStart, theEnd));

//call TimeDifference (theEnd, theStart). On return, print out the number of minutes

printf("Time from theEnd to theStart: %d minutes\n", TimeDifference(theEnd, theStart));

}

**Program Output:**  
theStart is HH:11 MM:52

theEnd is HH:14 MM:43

Time from theStart to theEnd: 171 minutes

Time from theEnd to theStart: 1269 minutes

**Problem 2 follows…**  
**Description:** Problem 2 is designed to demonstrate file input/output. It consists of a typical typedef and functions to successfully read a file, write to an array and display the array’s data.

**Program Code (main.c):**

#include <stdio.h>

#include <stdlib.h>

#include <assert.h>

//userDataStruct is a struct type that has 3 fields

typedef struct userDataStruct {

//array sized to hold 30 characters and a NULL (zero byte) at the end

char name[31];

int age;

int salary;

} userDataStructType;

//function to populate the array

int getData(char fileName[], userDataStructType userData[], int numItems) {

// attempt to open the file

FILE \* inputFile = fopen(fileName, "r");

// exit if the file did not open

assert(inputFile != NULL);

// the file opened, so proceed and process its contents

// read records from the file, printing out each record

int count;

int i = 0;

do {

// try to read in a record

count = fscanf (inputFile, "%s%d%d", userData[i].name, &userData[i].age, &userData[i].salary);

//increment a new numItems value

i++;

} while ((count == 3) && (i <= numItems));

fclose (inputFile);

//return new numItems value starting at 1

return i-1;

}

//function to show array data

int showData(userDataStructType userData[], int numItems) {

printf("\nThere are %d item(s) in the array:\n", numItems);

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

printf("%s %d %d\n", userData[i].name, userData[i].age, userData[i].salary);

}

return 0;

}

int main()

{

//initialize variables used for reading a file

int numItems;

char fileName[] = "Problem1\_2DataFile.txt";

//make array of userDataStruct using the typedef, fixed size at 10

userDataStructType userData[10];

//prompt user for number of people to get data for

printf("Give me a number from 1 to 10: ");

scanf("%d", &numItems);

//check that number is within range

if (numItems < 1 || numItems > 10) {

printf("Number is outside the range of 1 to 10. Exiting Program\n");

exit(0);

}

//populate array data and retrieve number of array items filled

numItems = getData(fileName, userData, numItems);

//display contents of the array

showData(userData, numItems);

}

**Data File:**  
Oren 22 88989

Dwayne 1 27352

Lynn 15 62583

Larry 28 95456

Quinn 58 46916

Carrol 10 56519

Von 45 73677

Ezra 23 56612

**Program Output (3):**

Give me a number from 1 to 10: 3

There are 3 item(s) in the array:

Oren 22 88989

Dwayne 1 27352

Lynn 15 62583

**Program Output (6):**

Give me a number from 1 to 10: 6

There are 6 item(s) in the array:

Oren 22 88989

Dwayne 1 27352

Lynn 15 62583

Larry 28 95456

Quinn 58 46916

Carrol 10 56519

**Program Output (10):**

Give me a number from 1 to 10: 10

There are 8 item(s) in the array:

Oren 22 88989

Dwayne 1 27352

Lynn 15 62583

Larry 28 95456

Quinn 58 46916

Carrol 10 56519

Von 45 73677

Ezra 23 56612

**Problem 3 follows…**  
**Description:** Problem 3 is designed to demonstrate building a program that has multiple files. It consists of a two .c files (one of which is the main) and a .h file.

**Output File:**

Kicking off the PingPong for 5 iterations

in Ping with value 5

in Pong with value 4

in Ping with value 3

in Pong with value 2

in Ping with value 1

in Pong with value 0

Back from the PingPong iterations.

**Problem 5 follows…**

**Description:** Problem 5 is designed to demonstrate swapping information passed into a function using pointers. It consists of two integers, a typical typedef struct and two functions which return nothing but swap caller’s data.

**Program Code (main.c):**

#include <stdio.h>

#include <stdlib.h>

//numStruct is a struct type that has 2 fields

typedef struct numStruct {

int a;

int b;

} numStructType;

//switch int values using pointers and return nothing

void Swap(int \*I, int \*J) {

//save one pointer int to a temporary int variable

int temp = \*I;

//set first int to second, then set second int to temp

\*I = \*J;

\*J = temp;

}

//switch struct int values using pointers and return nothing

void SwapStructs(numStructType \*ptr1, numStructType \*ptr2) {

//save one pointer struct to a temporary struct variable

numStructType temp = \*ptr1;

//set first struct to second, then set second struct to temp

\*ptr1 = \*ptr2;

\*ptr2 = temp;

}

int main()

{

//initialize variables

int I = 1;

int J = -1;

numStructType ptr1;

numStructType ptr2;

//set struct integers

ptr1 = (numStructType){.a = 10, .b = 20};

ptr2 = (numStructType){.a = 30, .b = 40};

//print int values before and after calling Swap

printf("Before call..I=%d, J=%d\n", I, J);

Swap(&I, &J);

printf("After call..I=%d, J=%d\n", I, J);

//print struct int values before and after calling SwapStructs

printf("Before SwapStructs..ptr1 contains %d and %d\n", ptr1.a, ptr1.b);

printf("Before SwapStructs..ptr2 contains %d and %d\n", ptr2.a, ptr2.b);

SwapStructs(&ptr1, &ptr2);

printf("After SwapStructs..ptr1 contains %d and %d\n", ptr1.a, ptr1.b);

printf("After SwapStructs..ptr2 contains %d and %d\n", ptr2.a, ptr2.b);

return 0;

}

**Program Output:**

Before call..I=1, J=-1

After call..I=-1, J=1

Before SwapStructs..ptr1 contains 10 and 20

Before SwapStructs..ptr2 contains 30 and 40

After SwapStructs..ptr1 contains 30 and 40

After SwapStructs..ptr2 contains 10 and 20