ASSESSMENT COVER SHEET

Student Name	Shane Reynolds
Student ID	262538
Assessment Title	Quiz 3
Unit Number and Title	HIT365
Lecturer/Tutor	Kai Wang
Date Submitted	16/5/2014
Date Received	16/5/2014

KEEP A COPY
Please be sure to make a copy of your work. If you have submitted assessment work electronically make sure you have a backup copy.

PLAGIARISM
Plagiarism is the presentation of the work of another without acknowledgement. Students may use a limited amount of information and ideas expressed by others but this use must be identified by appropriate referencing.

CONSEQUENCES OF PLAGIARISM
Plagiarism is misconduct as defined under the Student Conduct By-Laws. The penalties associated with plagiarism are designed to impose sanctions on offenders that reflect the seriousness of the University's commitment to academic integrity.

I declare that all material in this assessment is my own work except where there is a clear acknowledgement and reference to the work of others. I have read the University's Academic and Scientific Misconduct Policy and understand its implications.*

http://www.cdu.edu.au/governance/documents/AcademicandScientificMisconductPolicyv1.03Jan2011.pdf.

Signed……Date…13/01/2014……

*** By submitting this assignment and cover sheet electronically, in whatever form you are deemed to have made the declaration set out above.**

C PROGRAMMING – QUIZ 3

HIT 365 – C Programming

QUESTION 1

There are three integer variables `rock`, `paper` and `scissors` which have been initialized. Write code to swap the values in these variables around so that `rock` is given `paper`'s original value, `paper` is given `scissor's` original value and `scissors` is given `rock`'s original value.

Response

```c
/*
 * Question1.c
 *  Created on: 16/05/2014
 *  Author: Shane
 */

#include <stdio.h>
#include <stdlib.h>

void swap(int *ptr1, int *ptr2, int *ptr3);

int main(void){

        int rock = 1, paper = 2, scissors = 3;
        int *rockPtr, *paperPtr, *scissorsPtr;

        rockPtr = &rock;
        paperPtr = &paper;
        scissorsPtr = &scissors;

        printf("%d\n", *rockPtr);
        printf("%p\n\n", &rockPtr);
        printf("%d\n", *paperPtr);
        printf("%p\n\n", &paperPtr);
        printf("%d\n", *scissorsPtr);
        printf("%p\n\n", &scissorsPtr);

        swap(rockPtr, paperPtr, scissorsPtr);

        printf("%d\n", *rockPtr);
        printf("%p\n\n", &rockPtr);
        printf("%d\n", *paperPtr);
        printf("%p\n\n", &paperPtr);
        printf("%d\n", *scissorsPtr);
        printf("%p\n\n", &scissorsPtr);

        return 0;
}

void swap(int *ptr1, int *ptr2, int *ptr3){

        /* Store variable 1 address in temp1 */
        int temp1 = *ptr1;
        /* The object pointed to by ptr1 is assigned the value pointed to by ptr2 */
        *ptr1 = *ptr2;
        /* Store variable 2 in temp2 */
        int temp2 = *ptr3;
        /* The object pointed to by ptr3 is assigned the temp1 variable */
        *ptr3 = temp1;
        /* The object pointed to by ptr2 is assigned the temp2 variable */
        *ptr2 = temp2;


}
```

Hi Dr Kai

I know that I could have done this problem simply by using a temp variable and swapping the values of the variables this way, but I wanted to keep up my practice using pointers so I created a function which manipulates the pointers to the variable locations instead.

Sorry for the extended response.

Cheers

Shane.

QUESTION 2

There is a C array called `pair` containing two integers of unknown value.

> Write code to check if the elements are in descending order, and if they are - swap them. For example if `pair` initially contains [17, 8], after your code has executed it would contain `[8,17]`. If `pair` initially contains [12, 16], it would remain unchanged.

Response

```c
/*
 * Question2.c
 * Created on: 16/05/2014
 * Author: Shane
 */

#include <stdio.h>

void swapArray(int array[2]);

int main(void){

	/* Initialise the array pair */
	int pair[2] = {12, 16};

	/* Print off the initial values in array */
	printf("%d%7d\n", pair[0], pair[1]);

	/* Test to see if the first element of the array is more than the second */
	if (pair[0] > pair[1]){
		/* If so, then call the swap function */
		swapArray(pair);
		/* Print the swappped array */
		printf("%d%7d\n", pair[0], pair[1]);
	}else{
		printf("The array doesn't need to be changed.");
	}

	return 0;
}

void swapArray(int array[2]){

	/* Temporarily store the first element of the array */
	int temp = array[0];

	/* Reassign the array elements */
	array[0] = array[1];
	array[1] = temp;
}
```

QUESTION 3

There are three C arrays: `oldList` which contains a list of unknown values of type integer type; `newList` which is of the same length as `oldList` but contains only 0s; and `indexes` which contains integers representing a number of valid indexes to elements of `oldList`.

Now consider the following code:

```
int oldList[5] = {1, 4, 5, 7, 9};

int newList[5] = {0, 0, 0, 0, 0};
```

(a) What does `newList` contain after the above code has been executed?

```
newList contains 5 zeros in the array
```

(b) What does `oldList` contain after the above code has been executed?

```
oldList contains the numbers 1, 4, 5, 7 and 9
in the array
```

(c) Write in the box below a program that copies oldlist to newlist;

Response

```c
/*
 * Question3.c
 * Created on: 16/05/2014
 * Author: Shane
 */

#include <stdio.h>

int main(void){
	int oldList[5] = {1, 4, 5, 7, 9};
	int newList[5] = {0, 0, 0, 0, 0};
	int i, n = sizeof(oldList)/sizeof(oldList[0]);

	printf("%d\n\n", sizeof(oldList));

	/* Copy the old list into the new list */
	for (i = 0; i < n; i++){
		newList[i] = oldList[i];
	}

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

		/* Print the new list to check that it has done the job */
		printf("%d ", newList[i]);
	}

	return 0;
}
```

QUESTION 4

Suppose you had a C array of integers called `mirrors`. Write code that would print out every element of that array that had the same value as its index position. For example, given the array `{0, 2, 1, 3}`, the code would print the values: 0 and 3.

Response

```c
/*
 * Question4.c
 * Created on: 16/05/2014
 * Author: Shane
 */

#include <stdio.h>

int main(void){

	int mirrors[4] = {0, 2, 1, 3};
	int i;

	/* Dynamically determine the size of the mirrors array */
	int n = sizeof(mirrors)/sizeof(mirrors[0]);

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

		/* Test each element to see if it is the same as its index */
		if (mirrors[i] == i){
			/* Print array element if true */
			printf("%d\n", mirrors[i]);
		}
	}

	return 0;

}
```

QUESTION 5

In this question, you are given two pieces of code which are the same, except for one line which differs in the two listings. In the second piece of code, a box replaces the code that differs in that listing. Each code segment is to perform a different task, as explained below.

The following code prints all the positive ELEMENTs in a given array of integers called `nums` or `length NLEN`:

#define NLEN 33

```
for (int index = 0; index < NLEN     ; index++) {
    if (nums[index] > 0) {
        printf("%d",nums[index]);
    }
}
```

Complete the following code to print the INDEXES of all the positive elements in that list.

```
for (int index = 0; index < NLEN; index++) {
    if (nums[index] > 0) {
        printf(  "%d\n", index  );
    }
}
```

Response

```
/*
 * Question5.c
 * Created on: 16/05/2014
 * Author: Shane
 */

#include <stdio.h>
#define NLEN 5

int main(void){

      int index;
      int nums[33] = {-1, 2, -3 , -4, 10};

      for (index = 0; index < NLEN; index++){
              if (nums[index] > 0) {
                  printf("%d\n", index);
              }
      }

      return 0;

}
```