ICT1002 – LAB – WEEK 10

Pointers

1. OBJECTIVES

1. Understand and apply pointer concepts.
2. Understand and apply the relationship between pointers and arrays in C.
3. To understand and use user-defined data types.

2. EXERCISES FOR WEEK 10 LAB

WEEK_10_LAB_EXE_1: POINTER SYNTAX

Assume that the following lines of code have been executed.

```
int *zPtr; /* zPtr will reference array z */
int *aPtr = NULL;
void *sPtr = NULL;
int number, i;
int z[ 5 ] = { 1, 2, 3, 4, 5 };
zPtr = z;
sPtr = z;
```

Find the error in each of the following fragments of code to be executed after the code above. Check your answers by correcting the error and compiling it.

a)
```
/* use a pointer to get the first value of array */
number = zPtr;
```

b)
```
/* assign array element 2 (the value 3) to number */
number = *zPtr[ 2 ];
```

c)
```
/* print the entire array z */
for ( i = 0; i <= 5; i++ ) {
        printf( "%d ", zPtr[ i ] );
}
```

WEEK_10_LAB_EXE_2: POINTERS, ARRAYS, AND FUNCTIONS

Write functions that perform each of the following tasks, using the given function prototype. In each case, you may assume that the destination array is large enough to hold the data to be put into it.

a) Copy the contents of the array `src` (of length `n`) to the array `dest`:

```
void acopy(int *dest, const int *src, int n)
```

B) Compare the contents of two arrays of length `n`, returning 1 if the two arrays contain the same elements in the same order and 0 otherwise:

```
int acmp(const int *a1, const int *a2, int n)
```

3. WEEK_10_LAB_ASSIGNMENT: GUESSWORD

Write a program called `guessword` that plays a two-player word-guessing game using a similar procedure to the `guessint` program from Lab 8. The game will proceed as follows:

1. Player 1 will be asked to enter a word of up 12 letters. The word should contain only the lower-case English letters from ‘a’ to ‘z’, and no punctuation marks or digits.
    a. If Player 1 enters a word with upper case letters, the program should change them to lower case.
    b. If Player 1 enters a word with punctuation marks or digits, he or she should be asked to enter another word.
    c. The program does **not** need to check that the word is a “real” word (i.e. in a dictionary).
2. Player 2 (who again has not been watching Player 1) will be asked to guess one letter at a time.
    a. At the beginning of each round, the program will output a row of characters containing one underscore for every letter in the word to be guessed. If Player 2 has previously guessed a letter that is in the word, the underscore will be replaced by that letter.
    b. Player 2 will enter one letter. If he or she enters an upper-case letter, the program will convert it to lower case. If he or she enters a punctuation mark or digit, he or she will be considered to have made an incorrect guess.
    c. If the letter is **not** in the word, the number of incorrect guesses will be incremented.
    d. If the letter is in the word, every position in the word in which that letter occurs will be revealed at the start of the next round.
3. The game ends when either Player 2 has guessed all of the letters of the word, or when Player 2 has made seven incorrect guesses.

Some sample output is shown below, with the user input shown in red:

```
Player 1, enter a word of no more than 12 letters:
Topsy-turvy
Sorry, the word must contain only English letters.
Player 1, enter a word of no more than 12 letters:
Cat
Player 2 has so far guessed: _ _ _
Player 2, you have 7 guesses remaining. Enter your next guess:
e
Player 2 has so far guessed: _ _ _
Player 2, you have 6 guesses remaining. Enter your next guess:
a
Player 2 has so far guessed: _ a _
Player 2, you have 6 guesses remaining. Enter your next guess:
c
Player 2 has so far guessed: c a _
Player 2, you have 6 guesses remaining. Enter your next guess:
t
Player 2 has so far guessed: c a t
Player 2 wins.
```

Note the following:

- Use `#define` and comments as in Lab 8.
- You can use the `ctype.h` and `string.h` libraries for manipulating characters and strings.
- You may find it useful to write “helper” functions that perform tasks like checking that Player 1’s string is valid, whether and where Player 2 has correctly guessed a letter, and so on.