MIDDLE EAST TECHNICAL UNIVERSITY, NORTHERN CYPRUS CAMPUS
CNG213 Data Structures – Worksheet 1

Revision
(Pointers, Pointers and Arrays, and Structures)

Purpose:
This worksheet aims to revise the following topics: pointers, the relationship between pointers and arrays, and structures.
At the end of this worksheet, you will revise:
1. how to use pointers, and how to do pointer based C programming;
2. the relation between pointers and arrays;
3. how to do dynamic memory allocation;
4. how to use structures.

Task 1- Pointers:

1) Write a program that reads the radius of a sphere using a variable double type in the function main(). Pass the value of this variable to a function, which calculates the volume and surface area for the sphere. The function signature should be as follows:

```
void calculate_volume_surface(double radius, double *volume, double
*surface)
```

Note: $V = \frac{4}{3}\pi r^3$ $A = 4\pi r^2$

2) Write a program that includes the following function;

```
void find_largest_smallest(int a[], int n, int *largest, int *smallest)
```

When passed an array of length n, the function will search for its largest and smallest elements, storing it in the variable pointed by the largest and smallest. A sample run can be as follows:

```
How many elements you want to store (n)? 3
Enter a value to be stored: 10
Enter a value to be stored: 20
Enter a value to be stored: 3
Largest value stored is 20 and the smallest is 3.
```

3) Write a program that includes the following function that determines the smallest number of $20, $10, $5, and $1 necessary to pay the amount represented by the dollars parameter.

```
void pay_month(int dollars, int *twenties, int *tens, int *fives, int
*ones)
```

The twenties parameter points to the variable in which the function will store the number of $20 bills required. The tens, fives and ones parameters are similar. A sample run can be as follows:

```
Enter the amount to pay: 95
You need to pay 4x$20 + 1x$10 + 1x$5
```

4) A C program can represent a real polynomial p(x) of degree n as an array of the real coefficients $a_0,a_1,.....,a_n$ where a_n does not equal to 0

$p(x)= a_0+a_1x^1+a_2x^2+.....+a_nx^n$

write a program that inputs a polynomial of maximum degree 8 and then evaluates the polynomial at various values of x. Include a function get_poly that fills the array of coefficients and sets the degree of the polynomial, and a function eval_poly that evaluates a polynomial at a given value of x. Use these function prototypes:

```
void get_poly(double **coeff, int* degreep);
double eval_poly(double coeff[], int degree, double x);
```

Task 2- Arrays and Pointers:

5) Write a program that includes the following function:

```
int inner_product(int *a, int *b, int n)
```

a and b both point to arrays of length n. The function should return a[0]*b[0]+a[1]*b[1]+a[2]*b[2]+…+a[n-1]b[n-1]

Use pointer arithmetic-not subscripting-to visit array elements. A sample run can be as follows:

```
How many elements you want to store (n)? 3
Enter a value to be stored in the first array: 2
Enter a value to be stored in the first array: 3
Enter a value to be stored in the first array: 2
Enter a value to be stored in the second array: 3
Enter a value to be stored in the second array: 2
Enter a value to be stored in the second array: 3
Result of inner product is: 18
```

6) Write a program that includes the following function:

```
int search (int *a, int n, int key)
```

a is an array to be searched, n is the number of elements in the array and the key is the search key. Search should return true (1) if key matches some element of a, and false (0) if it doesn't. Use pointer arithmetic – not subscripting (e.g., [1]) - to visit array elements. A sample run can be as follows:

```
How many elements you want to store (n)? 5
Enter a value to be stored: 1
Enter a value to be stored: 6
Enter a value to be stored: 7
Enter a value to be stored: 10
Enter a value to be stored: 3
Enter a value to be searched: 5
This value does not exist
Search again (Y/N)? Y
Enter a value to be searched: 6
This value exist
Search again (Y/N)? N
```

7) A normalized vector X is defined as

$$x_i = \frac{v_i}{\sqrt{\sum_{i=1}^{n} v_i^2}} \quad i=1,2...n$$

Each element of the normalized vector X is computed by dividing the corresponding element (vi) of the original vector by the square root of the sum of the squares of all the original vector's elements. Design and test a program that repeatedly scans and normalized of vectors in different lengths. Define the following functions:

`Scan_vector (double *x, int size)`: Will read the data for a vector.
`Normalize_vector (double *x, double *v, int size)`: will normalize the data loaded in the previous function.
`Print_vector (double *x, double *v, int size)`: will display the content of the original and normalized vector.

Task 3- Dynamic Memory Allocation:

8) Write a program to read a list of numbers entered from the terminal. Before this list of numbers is entered, the program first prompts the user to specify how many numbers is going to be entered. Based on this, the memory will be allocated dynamically for an array, which will be used to store the data. Print the original entered data. Sort the data using a function called sort and also include a function that is called median to calculate the median of the given numbers. Function signatures would possibly be as follows:

```
void populate_array(int *data, int size)
void sort_array(int *data, int size)
int median(int *data, int size)
```

9) Write a program that asks the user for two words, dynamically allocates space for each of the words, and then intertwines them. For example:

```
Enter first word: Pajama
Enter second word: Party
Intertwined: PPaajratmya
```

Task 4- Structures:

10) The following structures are designed to store information about objects on a graphics screen:
struct point {int x, y;};
struct rectangle {struct point upper_left, lower_right;};
A point structure stores the value of x and y coordinates of a point on the screen. A rectangle structures stores the coordinates of the upper left and lower right corners of a rectangle. Write a program that includes functions that perform the following operations on a rectangle structure r passed as an argument:
i) Compute the area of r;
ii) Compute the center of r, returning it as a point value. If either x or y coordinates of the center isn't an integer, store its truncated value in the point structure.
iii) Move r by x units in the x direction and y units in the y direction, returning the modified version of r. (x and y are additional arguments to the function).

A sample run can be as follows: Your program should ask the user to enter the upper left and lower right coordinates and then should provide a menu for the functions listed above (i, ii, iii) so that the user can choose which one to process.

11) Define a structure type to represent a word list. The structure will contain one string component for the language of the words (e.g., English, Japanese, Spanish), an integer component that keeps track of how many words are in the list, and an array of MAX_WORDS 20-character strings to hold the words. Define the following functions to work with word lists:

    1. `load_word_list` - takes as parameters the name of an input file and a wordlist structure to be filled.
    2. `contains` - takes as parameters a word and a wordlist. If the word matches one of the wordlist entries, the function returns true, otherwise false.
    3. `add_word` - takes as parameters a word and a wordlist structure to modify. If the wordlist is already full, it displays the message "List full, word not added." If the word is already in the list, it leaves the structure unchanged. Otherwise, it adds the word to the list and updates the list size. Do not bother keeping the list in order.
    4. `equal_lists` - takes two wordlists as parameters and returns true if the lists are in the same language, have the same number of elements, and every element of one list is found in the other. (Hint: call contains repeatedly.)
    5. `display_word_list` -displays all the words of its wordlist structure parameter in four columns.

    Write a program that fills a wordlist from a data file. Then prompt the user to enter a language and 12 words to add to a different list. Then ask the user to enter some words to search for in the first list using contains, and print a message indicating whether each is found. Use equal_lists to compare the two lists, printing an appropriate message. Finally, use display_word_list to output each list.

Please note that from this worksheet, you need to submit the solution of the following question (12)

12) Write a data structure to represent Cars for BMW. In this structure, you need to keep track of the car model, year and the price. Define an array of 2 car structure that can be used to store information about the BMW cars. Write functions that perform the following operations:

    a) addCar() – This function will take the array of car structure created, and also the number of cars stored in the array, and add a new car to the array. Please note that initially, the size of the array is 2. If the number of cars is already 2, then the main function should reallocate the size of the array such that a new car can be added to this array.

    b) findCheapest() – This function takes the array of car structure created, the number of cars stored and finds the car with the lowest price.

    c) storeCars() – This function takes the array of car structure created and the number of cars, stores them to a file called "BMWcars.txt" with order (from newest to oldest year).

A sample run would be as follows:

```
BMW Cars

1) Add car
2) Find the cheapest car
3) Store cars and exit
What would you like to do? 1
Enter the model of the car: 2 Series
Enter the year of the car: 2015
Enter the price of the car: 26780

2 Series is added!

1) Add car
2) Find the cheapest car
3) Store cars and exit
What would you like to do? 1
Enter the model of the car: 1 Series
Enter the year of the car: 2019
Enter the price of the car: 26900

1 Series is added!

1) Add car
2) Find the cheapest car
3) Store cars and exit
What would you like to do? 2
Cheapest BMW car is 2 series 2015 26780£!

1) Add car
2) Find the cheapest car
3) Store cars and exit
What would you like to do? 3
BMWCars.txt is successfully created!
```