THE UNIVERSITY OF AUCKLAND

SECOND SEMESTER, 2022
Campus: City

Introduction to Engineering Computation and Software Development

Test 2 Question Booklet

(Time Allowed: NINETY minutes)

NOTE: This question booklet comprises 6 printed pages, including the cover page.

Answer ALL questions, entering your answers on the provided answer booklet, in the appropriately labelled space. Do not tear any page out from the answer booklet.

The test will be marked out of 50. There are 4 questions. Allocate your time carefully.

This test is restricted book (written upon). You may bring into the test **a single sheet of A4 paper** which can have handwritten and/or typed notes on both sides. Your sheet must not have anything stuck to it.

Calculators are permitted.

Question 1 (10 marks)

The *range* of a list of values is the difference between the smallest value and the largest value in the list. For example, given the following list of numbers:

123, 120, 118, 119, 121, 126, 127, 130, 129, 132

the range is 14 (which is the difference between the smallest, 118, and largest, 132, value).

Define a function called `Range()` which is passed two inputs: an array of integer values and the number of elements in the array. The function should calculate and return the range of these values. Below are several examples of the `Range()` function being called:

```
int valuesA[10] = {123, 120, 118, 119, 121, 126, 127, 130, 129, 132};
int valuesB[5] = {2, 6000, 55, -1, 20};
int valuesC[5] = {1, 2, 3, 4, 5};

printf("%d\n", Range(valuesA, 10));
printf("%d\n", Range(valuesB, 5));
printf("%d\n", Range(valuesC, 5));
```

The output from the code above is shown below:

```
14
6001
4
```

Write a definition for the `Range()` function in the answer booklet.

Be sure to indent your code consistently and include a short header comment. (10 marks)

Question 2 (12 marks)

The program shown on the next page is intended to calculate and display the average amount of rainfall for three datasets. The `AverageRainfall()` function takes one input: an array containing integer values representing the amount of rainfall per day. The array must contain a special value, -9999, which represents the end of the rainfall data. Any values in the array which are negative represent invalid data and should be ignored from the average calculation. If there are no valid rainfall data, then the function should return -1.0. The printed output should always be displayed with 2 decimal places.

The program currently calculates the average of the following datasets:

```
int rainA[6] = {2, 4, 5, 5, 5, -9999};
int rainB[6] = {2, 0, -1, 0, 3, -9999};
int rainC[4] = {-10, -1, -2, -9999};

printf("Rainfall A: %.2f\n", AverageRainfall(rainA));
printf("Rainfall B: %.2f\n", AverageRainfall(rainB));
printf("Rainfall C: %.2f\n", AverageRainfall(rainC));
```

The output from the program *should be* the following:

```
Rainfall A: 4.20
Rainfall B: 1.25
Rainfall C: -1.00
```

However, the `AverageRainfall()` function contains THREE bugs. Each bug is localised to a single line of code. **For each bug you must do the following**:

1. Identify the line number (shown in the left margin) of the bug
2. Describe the bug in plain English
3. Provide a fix for the bug by rewriting the code for that line number correctly

```c
#include <stdio.h>

double AverageRainfall(int values[])
{
    int sum = 0;
    int count = 0;
    int i = 0;
    while (values[i] == -9999) {
        if (values[i] > 0) {
            sum += values[i];
            count++;
        }
        i++;
    }
    if (count == 0) {
        return -1.0;
    } else {
        return (double)(sum / count);
    }
}
```

In the Answer Booklet, you should organise your answer for this question into three sections, one for each bug. For each bug, ensure you answer the three items listed above. (12 marks)

Question 3 (14 marks)

A string in C is an array of characters that is terminated by the null character. Define a function called `RemoveLetter()` which is passed two inputs: a string and a character. The function should modify the string by removing all occurrences of the character specified as input.

For example, consider the code below which creates three strings and then prints each string surrounded by quotation marks. The code then calls the `RemoveLetter()` function, providing each string as input, and removing the characters 'a', 'G', and ' '. Finally, it prints the strings again.

```
char wordA[100] = "abracadabra";
char wordB[100] = "ENGGEN131";
char wordC[100] = "the quick brown fox";

printf("WordA = \"%s\"\n", wordA);
printf("WordB = \"%s\"\n", wordB);
printf("WordC = \"%s\"\n", wordC);

RemoveLetter(wordA, 'a');
RemoveLetter(wordB, 'G');
RemoveLetter(wordC, ' ');

printf("WordA = \"%s\"\n", wordA);
printf("WordB = \"%s\"\n", wordB);
printf("WordC = \"%s\"\n", wordC);
```

The output from the code above is shown below:

```
WordA = "abracadabra"
WordB = "ENGGEN131"
WordC = "the quick brown fox"
WordA = "brcdbr"
WordB = "ENEN131"
WordC = "thequickbrownfox"
```

Write a definition for the `RemoveLetter()` function in the answer booklet.

NOTE: You may find it helpful to declare a temporary array in your `RemoveLetter()` function. If you choose to do this, you can assume that the input string will be less than 100 characters in length.

Be sure to indent your code consistently and include a short header comment. (14 marks)

Question 4 (14 marks)

The following algorithm can be used to compute the prime factorization of a number, *n,* where *n* is an integer greater than 1:

1. Let *n* be the number to factorise
2. Let *p* initially be 2
3. Can you divide *n* by *p* with no remainder?
   - If so: do the division (assigning the result to *n*) and write down *p* as output
   - If not: add 1 to the value of *p*
4. Go back to step 3 and repeat until *n* becomes 1

The algorithm above will generate the prime factors in numerical order from smallest to largest. For example, if *n* is 10005, the prime factors will be generated in the order: 3 5 23 29

Consider the program below which prompts the user to enter an input value and then calculates and displays the prime factors of that input number. Some of the code is missing. **The program must display the prime factors from *largest* to *smallest*** (i.e. in decreasing numerical order). The program uses a helper function, called `PrintArray()` to print the factors. You must ensure that the array passed to this function contains the factors in the appropriate order so that they are displayed from largest to smallest.

```
#include <stdio.h>

void PrintArray(int values[], int length)
{
        for (int i = 0; i < length; i++) {
                printf("%d ", values[i]);
        }
}

int main(void)
{
        int input, n, p, i, temp;
        int factors[100] = {0};
        int numFactors = 0;

        printf("Enter number to factorize: ");
        scanf("%d", &input);

        n = input;
        p = 2;

        ***You must complete the code that is missing from this section of the program.***

        PrintArray(factors, numFactors);

        return 0;
}
```

In the Answer Booklet, you only need to write the code for the missing section above – you do not need to copy the existing code. (14 marks)