



Beijing-Dublin International College



SEMESTER II FINAL EXAMINATION - 2020/2021

School of Computer Science

COMP1002J Introduction to Programming 2

Assoc. Prof. Chris Bleakley

Dr. Brett Becker *

Time Allowed: 120 minutes

Instructions for Candidates

Answers are to be supplied in this exam paper in the boxes provided below each question. Candidates should first complete their answers on rough paper then copy their final answers into the correct box. Candidates must complete all questions. The distribution of marks in the right margin give an approximate indication of the relative importance of each part of each question. In this exam, computer code is in `courier new` font and program output is shaded.

BJUT Student ID: _____

UCD Student ID: _____

I have read and clearly understand the Examination Rules of both Beijing University of Technology and University College Dublin. I am aware of the Punishment for Violating the Rules of Beijing University of Technology and/or University College Dublin. I hereby promise to abide by the relevant rules and regulations by not giving or receiving any help during the exam. If caught violating the rules, I accept the punishment thereof.

Honesty Pledge: _____ (Signature IN PINYIN/ENGLISH)

Instructions for Invigilators

Calculators are not permitted. This is a closed-book exam.

Rough-work paper is to be provided for candidates. Candidates must answer questions on this exam paper, in the boxes provided. Blank 'exam booklets' are not to be used unless extra space is needed.

Question 1:

Write a **complete** C program that uses a **file pointer** and the `fopen` function to open a file called `display.txt` for reading. The program should read the file `display.txt` one character at a time and print each character to the screen one at a time. The end result is that the file `display.txt` is copied/printed to the screen. Good programming practice should be followed. Your code should accomplish the task with clear, concise code and be easily readable.

[12 marks]

Question 2:

Explain what is meant by **statically allocated memory** in C. Be sure to use a diagram as part of your explanation.

[12 marks]

statically allocated memory is when the program first executed, the memory is created and the memory size not change.

Question 3: $f(n+1) = n f(n)$.

Write a **complete** C program with a **recursive** function that calculates the factorial of an integer called n. n should be initialised to the value 15. The function should return the answer and the main function should print this to the screen. Your answer should try to allow for large output.

[12 marks]

```
#include <stdio.h>

int recursive (int n) {
    if (n == 1 || n == 0) {
        return 1;
    }
    else {
        int m = n * recursive(n-1);
        return m;
    }
}

int main ( ) {
    int n = 15;
    int k = recursive (n);
    printf ("%d ", k);
}
```

Question 4:

Write a **complete** C program that calculates the factorial of an integer called **n** **iteratively** using a **function**. **n** should be initialised to the value 15. The function should return the answer and the main function should print this to the screen. Your answer should try to allow for large output.

[12 marks]

Question 5:

Write a **complete** C program that **defines a type** called Building. A Building is a **structure** that should contain:

- An integer called number_of_floors
- A double called area

The main function should declare a variable of type Building, assign the number_of_floors variable the value 4, and assign the area variable the value 250000.50. The main function should then print the number of floors and the area on separate lines. The area should be printed with two decimal places only.

[12 marks]

```
#include <stdio.h>

typedef struct {
    int    number_of_floors;
    double area;
} Building;

int main() {
    struct Building a = {4, 250000.50};
    printf("%d\n %.2lf", a.number_of_floors,
           a.area);
}
```

Question 6:

What is the output of the following program?

```
#include <stdio.h>
int main() {
    int x = 5;
    int *ptr = &x;
    printf("%d\n", *ptr);
    *ptr = 10;
    printf("%d\n", x );
    printf("%d\n", *ptr);
}
```

[10 marks]

5
5
10

Question 7:

The following C program is missing the function func.

```
#include <stdio.h>
//missing function func

int main () {
    double A[ 5 ] = {1, 2, 3, 4, 5};
    double B[ 7 ] = {7, 6, 5, 4, 3, 2, 1};
    int length_a = 5;
    int length_b = 7;
    double sum = func( length_a, A );
    printf ( " sum = %f\n", sum );
    sum = func( length_b, &B[0] );
    printf ( " sum = %f\n", sum );
}
```

The output of the program should be:

```
sum = 15.000000
sum = 28.000000
```

Write the code for the function func so that the program compiles with the indicated output.

[10 marks]

```
double func(int length_num, double num){
    int sum = 0;
    for(int n = 0, n < length_num, n++) {
        sum = sum + num[n];
    }
    return sum;
}
```

Question 8:

Explain what is meant by **dynamically allocated memory** in C. Explain how malloc, free, and realloc are used.

[10 marks]

dynamically allocated memory is a program to get memory from heap. and need the function such as

malloc, calloc to get memory and use free to reallocate memory after dynamically allocated memory, we can use realloc() to change the size of memory we getted.

So that we can allocate and reallocate memory before the function finished.

*intp = malloc(4 * sizeof(int)); and use byte-
and return a void point; need one
parameter which
is the size
of memory we need

free(*intp) which function need
one parameter which is the allocate memory
on heap

*newintp = realloc(*intp, 40 * sizeof(int))

the parameter one is old point where the memory allocated
and one is the new size we need. if the memory after old
memory is enough. will return the same point

if the memory after old memory have another used, will return a new point and have a new block, it don't have enough memory (continue) will return NULL point.

BDIC

Semester Two

Academic Year (2020 – 2021)

Question 9:

What is the output of the following code?

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

void a() {
    int i;
    for (i=0; i < 5; i++){
        printf("S%d ", i);
    }
}

void b() {
    int *i = malloc(sizeof(int));
    for (*i=0; *i < 5; (*i)++){
        printf("D%d ", *i);
    }
    free(i);
}

int main() {
    a();
    b();
}
```

[10 marks]

S₀ S₁ S₂ S₃ S₄ D₀ D₁ D₂ D₃ D₄