The University of Nottingham Ningbo China

SCHOOL OF COMPUTER SCIENCE

A LEVEL 1 MODULE, AUTUMN SEMESTER 2019-2020

PROGRAMMING AND ALGORITHMS (COMP1038)

Time allowed: 1.0 hour (60 minutes)

Candidates may complete the front cover of their answer book and sign their desk card but must NOT write anything else until the start of the examination period is announced.

Answer ALL questions.

No calculators are permitted in this examination.

Dictionaries are not allowed with one exception. Those whose first language is not English may use a standard translation dictionary to translate between that language and English provided that neither language is the subject of this examination. Subject specific translation dictionaries are not permitted.

No electronic devices capable of storing and retrieving text, including electronic dictionaries, may be used.

DO NOT turn examination paper over until instructed to do so.

ADDITIONAL MATERIAL: None.

INFORMATION FOR INVIGILATORS: Collect both the exam papers and the answer booklets at the end of exam.

Turn Over

Question 1: Explain the difference between the *i++* and *++i* operations in C. (2 marks)

Question 2: Explain what a file pointer is and declare a file pointer in C. (2 marks)

Question 3: What is a constant pointer? What is a pointer to constant value? Declare an int type constant pointer ‘ptr1’ and an int type pointer to constant value ‘ptr2’ in C. (3 marks)

Question 4: Write a *structure* definition, an alias and an example declaration in C to store a student’s information. The structure must contain *ID*, *student_name,* and *module_marks*. The *ID* variable must be able to store a whole number. The *student_name* variable must be able to store 50 characters. The *module_marks* must be able to store the marks from 8 modules, these marks may contain decimal points.

Create an alias for this structure calling *Records,* and declare an array of 1000 elements of this structure type using the alias. (3 marks)

Question 5: Complete the function body of *check* in the program below. This function should take two arguments, an integer called *input* and an integer pointer called *ptr*. A calculation to determine whether the *input* is odd, even or zero must be done within this function. If *input* is zero, the program should print 0. If *input* is odd, the program prints -1. If *input* is even, the program prints 1. You can assume that the user will only enter a single whole number at each prompt (ie, no text, no floating-point numbers, no empty input, etc) (3 marks)

```
#include <stdio.h>

void check(int input, int *ptr);

int main(void)
{
        int in = 0;
        int out = 0;

        while(in != -9999)
        {
                scanf("%d", &in);
                check(in, &out);
                printf("%d\n", out);
        }
        return 0;
}
```

Question 6: Point out a potential mistake contained in the function 'func' and suggest how to correct it. (4 marks)

```
#include <stdlib.h>
struct ex
{
        int i;
        float j;
        char *s;
};

void func (void)
{
    struct ex *p = malloc(sizeof(struct ex));
    p->s = malloc(20 * sizeof(char));
    free(p);
}
```

Question 7: Why do we need to use a pointer to pointer as an argument of a function? Write an example program of this situation. (4 marks)

Question 8: What is the output of the following program? (4 marks)

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

int f1( int x, int y)
{
          x=x+2;
          y=y+3;
          return x+y;
}
```

```
int f2(  int *x, int y)
{
        *x=*x+2;
        y=y+3;
        return *x+y;
}

int f3 ( int *x, int *y)
{
        *x=*x+2;
        *y=*y+3;
        return *x+*y;
}

int f4( int x,  int *y, int *z)
{
        x=x+*y;
        *y=*z+3;
        z=&x;
        *z=*y*2;
        return *z;
}

int main(int argc, char *argv[])
{
        int k=2, m=1, r=3;
        printf("1) %d %d %d \n", k, m, r);
        r=f1(k, m);
        printf("2) %d %d %d \n", k, m, r);
        r=f2(&k, m);
        printf("3) %d %d %d \n", k, m, r);
        r=f3(&k, &m);
        printf("4) %d %d %d \n", k, m, r);
        r=f4(k, &m, &r);
        printf("5) %d %d %d \n", k, m, r);
        return 0;
}
```

End of exam questions.