The University of Nottingham Ningbo China

SCHOOL OF COMPUTER SCIENCE

A LEVEL 1 MODULE, AUTUMN SEMESTER 2018-2019

PROGRAMMING AND ALGORITHMS

Time allowed: **ONE 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.

Only silent, self-contained calculators with a Single-Line Display 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 your examination paper over until instructed to do so

INFORMATION FOR INVIGILATORS: Exam papers must be collected at the end of the exam.

Question 1: Write a declaration in C using struct to store a student's information. The struct must contain *ID_number*, *student_name*, and *final_mark*. The name variable must be able to store at most 50 characters. The *final_mark* must be able to store the marks from 6 modules. The mark may contain decimal points.

(3 marks)

Question 2: A common way to declare the main function is *int main(int argc, char *argv[])*. Explain what the arguments *argc* and *argv* represent using the following command line as an example. You must clearly indicate which value is stored where.

$ ./hello 1 test

(3 marks)

Question 3: Write a complete program which contain a function called *foo*. This function should take an integer as a parameter. If the integer is greater or equal to 2, the function should return the number of integer *factors* that the number has, including 1 or itself. Otherwise, it should return -1. A *factor* is a number which divides the other number with no remainder. For example, 16 has 5 factors (1, 2, 4, 8, 16), 9 has 3 factors (1, 3, 9). The output from *foo* should be displayed from the main function.

(3 marks)

Question 4: Explain binary search and its requirements. Write a **declaration** of a function *binary_search*. The function should take an integer array and a given value and should return a value to show whether or not the given value is contained in the array.

(2 marks)

Question 5: A programmer has written a program in C that asks the user to enter a whole number between 0 and 10 (inclusive of both). The program reads the user's input as a string then converts it to an integer. Write the different string values you would use to test the program works correctly and justify why you chose them. Choose your values carefully and write only as many as you think you require.

(3 marks)

Question 6: The following function takes an integer array and the length of that array. It should calculate the sum of that array and return the computed value. Identify and explain the two errors in this function and explain how to fix them.

(2 marks)

```
#include <stdio.h>
#include <stdlib.h>
int sum(int nums[], int nums_size)
{
        int *total = NULL;
        total = malloc(sizeof(int));
        if(total = NULL)
                exit(1);
        *total = 0;
        for(int i = 0 ; i < nums_size; i++)
        {
                *total += nums[i];
        }
        int* result = total;
        free(total);
        return *result;
}
```

Question 7: Briefly explain what undefined behaviour is in the C programming language. Under what circumstance(s) would calling the following C function result in undefined behaviour? (2 marks)

```
int divide(int *a, int *b)
{
        return *a  / *b;
}
```

Question 8: Below is part of a program implementing doubly linked list. Function push() creates the first node if the original list is empty, or pushes a new node at the beginning of an existing list. Point out two errors in the code and explain how to fix them.

(2 marks)

```
/* Node of a doubly linked list */
struct Node {
      int ID;
      char* name;
      struct Node* next; // Pointer to next node
      struct Node* prev; // Pointer to previous node
};

void push(struct Node** head_ref, int new_ID, char* new_name)
{
   struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
   if(new_node == NULL)
       return;

   new_node->ID = new_ID;
   new_node->name = new_name;

   new_node->next = head_ref;
   new_node->prev = NULL;

   if ((*head_ref) != NULL)
       (*head_ref)->prev = new_node;

   (*head_ref) = new_node;
   free(new_node);
}
```

Question 9: A segmentation fault is a common run-time error for C programs. Use examples (several lines of C code for each cause) to show two causes of this problem.

(2 marks)

Question 10: Write the output of the following program:

(3 marks)

```
#include <stdio.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)
{
        int *p;
        p = x;
        x = y;
        y = p;
        return *x + *y;
}

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

END