The University of Nottingham Ningbo China

SCHOOL OF COMPUTER SCIENCE

A LEVEL 1 MODULE, AUTUMN SEMESTER 2021-2022

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.

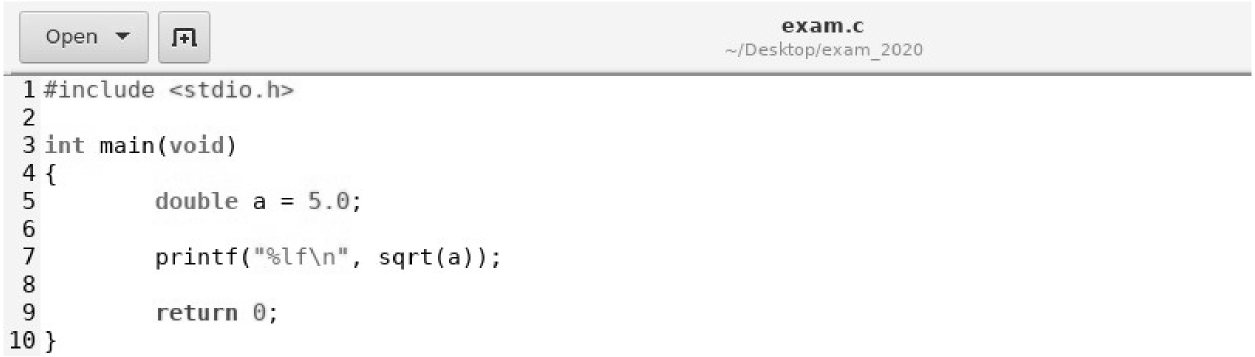
ADDITIONAL MATERIAL: None.

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

Question 1: A segmentation fault is a run-time error. Explain why a segmentation fault may occur in running C programs. (2 marks)

Question 2: What does dynamic memory allocation mean? What are functions in C programming to implement dynamic memory allocation? Use C code to show how to allocate and release memory dynamically for an integer array that contains 1000 elements. (2 marks)

Question 3: Explain what would happen when the program exam.c, as shown in Figure 1, is compiled using the command as shown in Figure 2 on cslinux. (2 marks)



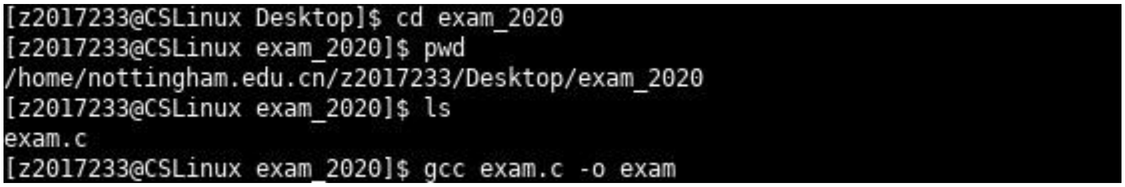
```c
#include <stdio.h>

int main(void)
{
        double a = 5.0;

        printf("%lf\n", sqrt(a));

        return 0;
}
```

Figure 1: exam.c



```
[z2017233@CSLinux Desktop]$ cd exam_2020
[z2017233@CSLinux exam_2020]$ pwd
/home/nottingham.edu.cn/z2017233/Desktop/exam_2020
[z2017233@CSLinux exam_2020]$ ls
exam.c
[z2017233@CSLinux exam_2020]$ gcc exam.c -o exam
```

Figure 2: Compiling on cslinux

Question 4: Use while loop to implement the same output as the following code. (2 marks)

```
for(a=1; a<=100; a++)
        printf("%d\n", a*a);
```

Question 5: Define a singly linked list to store several students' information. The name of the linked list should be *Student_list*. It should 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. (3 marks)

Question 6: Point out mistakes contained in the below program and suggest how to correct it. (4 marks).

```
#include <stdio.h>

struct emp
{
        char n[20];
        int age;
};

int main(void)
{
        struct emp e1 = {"David", 23};
        struct emp e2;
        scanf("%s %d", e2.n, e2.age);
        if(structcmp(e1, e2) == 0)
                printf("Equal structs.");
        else
                printf("Unequal structs.");
}

structcmp (struct emp x, struct emp y)
{
        if(x.n == y.n)
                if(x.age == y.age)
                        return 0;
        return 1;
}
```

Question 7: 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 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.