Family Name:. . . . . **Other Names:** . . . .

Student ID:. . . . . **Signature** . . . .

NWEN 241: Test 2

2024, December 11 ** WITH SOLUTIONS **

Instructions

- Time allowed: **90 minutes**
- Attempt **all** the questions. There are **32 marks** in total.
- Write your answers in this exam paper and hand in all sheets.
- If you think a question is unclear, ask for clarification.
- You may use unmarked paper Chinese-English translation dictionaries.
- You may write notes and workings on this paper, but make sure your answers are clear.

Questions	**Marks**	
1. True or False	[10]	
2. Multiple Choice	[10]	
3. Short Answer	[12]	
	TOTAL:	

Question 1. True of False **[10 marks]**

For the following statements, circle "true" or "false" for each statement.

(a) **[1 mark]** Memory allocated using `calloc` can be firstly released using `free` and then resized using `calloc` again.

true [✓ false]

(b) **[1 mark]** The dynamically allocated memory is stored in the heap segment.

[✓ true] false

(c) **[1 mark]** If the dynamically allocated memory is smaller than required, C language will print an error and crash.

true [✓ false]

(d) **[1 mark]** In a singly-linked list, the struct node is a built-in implementation in C language and it can be included from <stdlib.h>.

true [✓ false]

(e) **[1 mark]** A singly-linked list cannot be traversed backwards starting from the tail.

[✓ true] false

(f) **[1 mark]** When a program writes to the `stdout` stream, which is connected to the screen, the program immediately displays every character inputted by the user from the keyboard.

true [✓ false]

(g) **[1 mark]** The call rewind(fp) is equivalent to the call fseek(fp, -s, SEEK_CUR), where s is the size of the file (in bytes).

true [✓ false]

(h) **[1 mark]** The `exec()` system call does not return to the caller upon success.

[✓ true] false

(i) **[1 mark]** UDP sockets use the connectionless communication model.

[✓ true] false

(j) **[1 mark]** In a client-server model, the client and server must know each other's addresses before establishing a connection.

true [✓ false]

Question 2. Multiple choice ☑ **[10 marks]**

Hint: There might be more than one correct answer for each question

(a) **[1 mark]** What is the purpose of the `realloc` function in C?

- ☐ Allocating memory.
- ☐ Releasing dynamically allocated memory.
- ☐ Initializing memory to zero.
- ☐ Resizing dynamically allocated memory. ☑

(b) **[1 mark]** Which of the following is equivalent to `calloc(8,sizeof(double))` ?

- ☐ `malloc(8*sizeof(double))` ☑
- ☐ `malloc(8)`
- ☐ `malloc(sizeof(double))`
- ☐ `malloc(8, sizeof(double))`

(c) **[1 mark]** Select ALL valid statements about memory leak from the following statements:

- ☐ Program will not be able to access leaked memory. ☑
- ☐ Leaked memory will no longer be in the heap segment.
- ☐ Leaked memory cannot be freed, potentially causing program memory usage to keep on growing. ☑
- ☐ Leaked memory is automatically freed using garbage collection.
- ☐ Every instance of memory leak will always result in undefined program behaviour.

(d) **[1 mark]** Which stream buffering mode is used if reading or writing occurs as quickly as possible?

- ☐ Unbuffered ☑
- ☐ Line buffered
- ☐ Fully buffered
- ☐ Free buffered

(e) **[1 mark]** Select ALL valid reasons for a file opening failure.

- ☐ File is already opened. ☑
- ☐ File opened for reading does not exist. ☑
- ☐ File is empty.
- ☐ File cannot be accessed due to insufficient permissions. ☑
- ☐ File is already closed.

(f) **[1 mark]** Consider the following C code snippet:

```c
int* createArray(int size) {
    int arr[size];
    for (int i = 0; i < size; ++i) {
        arr[i] = i*i;
    }
    return &arr;
}
```

What is the potential issue with the code, and how can it be addressed?

- ☐ The function returns a pointer to a local array, leading to undefined behavior. ☑
- ☐ We need to use dynamic memory allocation instead of arrays. ☑
- ☐ Memory leakage is created from ++i.
- ☐ The issue can be resolved by using the `free` function.

(Question 2 continued on next page)

(Question 2 continued)

(g) **[1 mark]** Consider the following C code snippet for reading data from a file:

```
#include <stdio.h>

int main() {
    FILE *file = fopen("data.txt", "r");
    if (file != NULL) {
        int value;
        while (fscanf(file, "%d", &value) != EOF) {
            // What does this loop do?
        }
        fclose(file);
    }
    return 0;
}
```

What does the loop inside the `if` statement do, and how does it terminate?

- ☐ The loop reads integers from the file until the end of the file is reached. ☑
- ☐ The loop terminates if an error is encountered. ☑
- ☐ The loop reads integers from the file until a specific value is encountered.
- ☐ The loop reads characters from the file until a specific character is encountered.

(h) **[1 mark]** What is the correct description of the `exit()` system call in C?

- ☐ Enables explicit call for normal termination. ☑
- ☐ Cleans up and releases resources. ☑
- ☐ The exit status is an integer value from 0 to 256.
- ☐ By convention, the exit status 0 suggests abnormal termination.

(i) **[1 mark]** Consider the following C code snippet for creating a TCP server socket:

```
#include <stdio.h>
#include <sys/socket.h>

int main() {
    int serverSocket = socket(AF_INET, SOCK_STREAM, 0);
    if (serverSocket == -1) {
        // What does this condition check for?
    }
    return 0;
}
```

What does the condition `(serverSocket == -1)` check for, and what action is typically taken if this condition evaluates to true?

- ☐ The condition checks if the server socket was successfully created. ☑
- ☐ The condition checks if the socket is in TCP mode.
- ☐ If true, it indicates a failure to create the server socket, and appropriate error handling is needed. ☑
- ☐ The condition checks if the socket is in streaming mode.

(Question 2 continued on next page)

(Question 2 continued)

(j) **[1 mark]** A POSIX-based system include:

- ☐ Unix ☑
- ☐ Linux ☑
- ☐ Mac OS ☑
- ☐ Windows

Student ID: ....

Question 3. Short Answer questions **[15 marks]**

(a) **[6 marks]** Consider the following C code snippet:
(Hint: `atoi()` Convert string to integer)

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

int main(int argc, char* argv[]) {
    if (argc == 0)
        printf("hello");
    else if (argc == 1)
        printf("XMUT");
    else if (argc == 2)
        printf("OK");
    else if (argc >= 3) {
        int a = atoi(argv[1]);
        int b = atoi(argv[2]);
        printf("Value: %d\n", a - b);
    }
    return 0;
}
```

Suppose the code is compiled to an executable file named `thisProgram`.

i. **[2 marks]** What is the output if the file is executed as `./thisProgram`?

XMUT

ii. **[2 marks]** What is the output if the file is executed as `./thisProgram 5 3`?

Value: 2

iii. **[2 marks]** What is the output if the file is executed as `./thisProgram "5 3"`?

OK

(Question 3 continued on next page)

(Question 3 continued)

(b) **[2 marks]** Consider the following declaration:

```
#include <stdio.h>
int main() {
    FILE *file = fopen("numbers.txt", "r");
    int num1, num2, num3;
    fscanf( file , "%d %d %d", &num1, &num2, &num3);
    printf ("%d\n", num1 * num2 + num3);
    fclose ( file );
    return 0;
}
```

If the `numbers.txt` file contains `3 6 7 2` and it is opened successfully, what will be the output?

```
25
```

(c) **[4 marks]** Consider the following declaration:

```
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
int main() {
    pid_t pid = fork();
    if (pid == 0) {
        printf("Child\n");
        pid_t pid2 = fork();
        if (pid2 > 0) wait(NULL);
        printf("Child's child\n");
    }
    else if (pid > 0) {
        wait(NULL);
        fork();
        printf("Parent\n");
    }
    return 0;
}
```

What is the output of the following code? (Assume that fork() is successful)

```
Child
Child's child
Child's child
Parent
Parent
```

* * * * * * * * * * * * * * * *