Anytime before the actual exam
~~**Your real exam is 8/9 Feb!**~~
Expected Duration: 1 hour 30 minutes (expected in 2022, to confirm)
Time Allowed: 3 hours (expected in 2022, to confirm)
~~**Timed exam within 24 hours**~~

DEGREE of BSc Software Engineering (Graduate Apprenticeship)

SYSTEMS PROGRAMMING COMPSCI 2030

(Answer all 4 questions. This is a mock exam, v20220123)

~~**This examination paper is an open book, online assessment and is worth a total of 90 marks.**~~

1. These multiple-choice questions are negatively marked. For each correct answer you gain three marks, and for each wrong answer, you lose one. There is one correct answer per question.

There are 30 total marks in this question.

(a) Which of the following would require the most memory in standard C?

1. **`float`**
2. **`long int`**
3. **`char`**
4. **`double`**

[3]

(b) How are new types defined in C?

1. We use the typedef keyword
2. Types cannot be defined, only imported
3. We use the deftype keyword
4. C doesn't allow new types

[3]

(c) Which of the following is true about C pointers?

1. C pointers only come from `malloc()`, `calloc()`, and `realloc()`
2. Functions cannot return pointers
3. Pointer size depends on the system running our code
4. Any pointer returned by a function must be **`void`**

[3]

(d) We cannot allocate heap memory using:

1. `malloc()`
2. `calloc()`
3. `realloc()`
4. `free()`

[3]

(e) Stack memory is:

1. Unsafe to return the address of from a function
2. Allocated using `malloc()`
3. A method used by the operating system to allocate memory in pages
4. A modern memory management technique for deallocating space automatically, which C does not have

[3]

(f) A use-after-free error is caused by:

1. `free()`-ing memory twice
2. Accessing a value at a previously `free()`-d address
3. Losing reference to a pointer to heap-allocated memory
4. Running out of space to store values in memory

[3]

(g) Assume `arr` is an array of **`int`**s, and `index` is a valid integer index in `arr`. One of the following snippets has a different value to the others. Which one?

1. `index[arr]`
2. `arr[index]`
3. `(index + arr)`
4. `*(index + arr)`

[3]

(h) A race condition can occur when:

1. Two or more threads wait indefinitely on a lock held by another
2. Two or more threads access the same memory location at the same time
3. Two or more threads return a value at the same time
4. Two or more threads process copies of the same values

[3]

(i) Which of the following is true of threads?:

1. A thread is created and managed by an operating system
2. A thread is independent of the process which started it
3. A thread has heap memory separate to that of other threads
4. A thread is a language construct managed by a runtime, rather than the operating system itself

[3]

(j) `pthread_join()` is a function used to:

1. Signal that a thread should terminate
2. Signal the most important thread to the operating system, for scheduling purposes
3. Ensure safe access in a critical region
4. Wait for a thread to finish executing

[3]

2. These questions will center around code comprehension. Each snippet of code is completely self-contained, meaning that what you read should compile without warnings or errors without additions. However, in each question there will be an error regarding the use of C, an error pertaining to memory, or an error caused by mismanaged concurrency. For each question, you should:

1. Identify the bug in the code, and
2. Suggest a fix.

There are 21 total marks in this question.

(a)

```c
#include <stdio.h>

typedef struct _book {
    char title[50];
    int timesRead;
} book;

void printBook(book *toPrint) {
    printf("You␣have␣read␣%s␣%d␣times.\n",
           toPrint.title, toPrint.timesRead);
}

int main() {
    book goodBook = {"House␣of␣Leaves", 0};
    goodBook->timesRead++;
    goodBook->timesRead++;
    printBook(&goodBook);
    return 0;
}
```

[7]

(b)

```c
#include <stdio.h>

void calc_next_collatz(int *num) {
    if (num % 2 == 0) {
        num /= 2;
    } else {
        num = num * 3 + 1;
    }
}

int main() {
    int x = 6;
    while (x != 1) {
        calc_next_collatz(&x);
        printf("␣->␣%d", x);
    }
    printf("\n");

}
```

[5]

(c)

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


void * printInfo(void * arg) {
    char * user_input = (char *) arg;

    long int len = strlen(user_input);
    printf("Argument␣%s␣has␣length␣%ld\n", user_input, len);

    pthread_exit(&len);
}

int main(int argc, char * argv[]) {

    pthread_t * threads = malloc(argc * sizeof(pthread_t));
    int i;
    for (i = 0; i < argc; i++) {
        pthread_create(&threads[i], NULL, printInfo, argv[i]);
    }

    long int ** lengths = malloc(argc * sizeof(int *));

    for (i = 0; i < argc; i++) {
        pthread_join(threads[i], (void *)&lengths[i]);
    }

    long int total = 0;
    for (i = 0; i < argc; i++) {
        total += *lengths[i];
    }
    printf("\n\nTotal␣length␣is␣%ld\n", total);

    return 0;
}
```

[7]

CONTINUED OVERLEAF

3. These questions require longform answers. There are 23 total marks available.

(a) (i) Briefly explain the difference between stack and heap memory. Explain whether each one is specific to a thread, a process, both, or neither.

[4]

(ii) Can a memory leak happen to stack-allocated memory? Why, or why not?

[2]

(iii) One of the design decisions in C++ which diverges from C's design is its use of a technique called RAII. What does this acronym stand for, and in what way does it help memory management in C++?

[4]

Ignore, not covered!

(b) When freeing memory, fragmentation can occur, where small chunks of memory are left unused, but cannot be allocated because of their size and proximity to allocated blocks of memory. What operating system feature mitigates memory fragmentation?

[2]

Ignore, not covered!

(c) Briefly explain what POSIX is, why one might write POSIX-compliant code, and an example of when an operating system being non-POSIX-compliant impacts code portability.

[6]

(d) (i) What causes a race condition?

[2]

(ii) Race conditions can be defended against using a concurrency primitive which, when used incorrectly, causes a deadlock. What is the name of this concurrency primitive, and how can its misuse cause a deadlock?

[3]

4. Explain the significance of each line of the program on the following page commented with `???`. What do `i` and `c` do? What will this code print when run? This question is worth 16 marks.

CONTINUED OVERLEAF

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

typedef struct _n {
    int val;
    struct _n * l;
    struct _n * r;
} n;

void c(n * p, int v, int r) {
    n * new = malloc(sizeof(n)); // ??? 1
    new->l = NULL;
    new->r = NULL;
    new->val = v;
    if (r) { // ??? 2
        p->r = new;
    } else {
        p->l = new;
    }
}

int i(n * r, int e) {
    if (e == r->val) { // ??? 3
        return 0;
    } else if (e < r->val) {
        if (r->l == NULL) {
            c(r, e, 0); // ??? 4
            return 1;
        } else {
            return i(r->l, e); // ??? 5
        }
    } else {
        if (r->r == NULL) {
            c(r, e, 1);
            return 1;
        } else {
            return i(r->r, e);
        }
    }
}

int main() {
    n r = {5, NULL, NULL};
    i(&r, 2);
    i(&r, 5);
    i(&r, 8);
    i(&r, 3);
}
```

[16]