TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

EXAMINATIONS – 2017

TRIMESTER 1

NWEN241

SYSTEMS PROGRAMMING
VEC MOCK EXAM

Time Allowed: TWO HOURS

CLOSED BOOK

Permitted materials: No calculators are allowed.

No electronic dictionaries are allowed.

Paper foreign to English language dictionaries are allowed.

Instructions: The examination contains 5 questions. You must answer ALL questions.

The exam consists of 120 marks in total, with 20 marks for each of the 5 questions:

Question 1 C General Questions	**[20 marks]**
Question 2 Arrays Strings and Pointers	**[20 marks]**
Question 3 Data Structures and Memory	**[20 marks]**
Question 4 Projects, File I/O and Process Management	**[20 marks]**
Question 5 Python Fundamentals	**[20 marks]**

Student ID: .....

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Question 1. C General Questions **[20 marks]**

(a) **[4 marks]** Explain the four steps of compilation for C programs.

(b) **[4 marks]** Explain how the Stack, Heap and Data Segment sections are used in program memory and how these sections relate to compile-time or run-time memory allocation.

(c) **[4 marks]** The floating point format (IEEE 754 single-precision form) consists of three sections. Name each section and how it is used to construct the decimal number.

(d) **[4 marks]** The following macro computes the square of x. Discuss the issues with this macro:

```
#define SQ(x) x*x
```

(e) **[4 marks]** What does the following print out:

```
#include <stdio.h>

int main(void)
{
    int i;
    float k;

    for (k=i=6; i-->2; k+=0.5)
    {
        if (i%2)
            ;
        else
            printf("k=%.2f\n", k);
    }
    return 0;
}
```

Question 2. Arrays, Strings and Pointers **[20 marks]**

(a) **[6 marks]** Consider the following code:

```
int a[3] = {1, 2, 3};
int *pa = a;

int m[4][4] = {{2, 4, 6, 8}, {22, 44, 66, 88},
               {1, 3, 5, 7}, {11, 33, 55, 77}};
int (*pm)[4] = m;
```

Give the outputs of the following `printf` statements.

`printf("%d", *a);` **[1 mark]**

`printf("%d", *(a+1));` **[1 mark]**

`printf("%d", pa[1]);` **[1 mark]**

`printf("%d", *(*m+2));` **[1 mark]**

`printf("%d", *(m[1]+2));` **[1 mark]**

`printf("%d", (*(pm+3))[2];` **[1 mark]**

(b) **[4 marks]** Give a declaration for the variable `p` in each of the following cases.

`p` is an array of 5 elements of pointer to `char` **[1 mark]**

`p` is a pointer to an array of 10 elements of `float` **[1 mark]**

`p` is a `const` pointer to `int` **[1 mark]**

`p` is a function that takes a `const int` and a `float` array as arguments and returns a pointer to a `const int` **[1 mark]**

(c) **[2 marks]** Consider the following code:

```
char s1[] = "Hello";
char s2[] = {'H', 'e', 'l', 'l', 'o'};
```

`sizeof(s1)` does not equal `sizeof(s2)`. Discuss why this is.

(d) **[3 marks]** Consider the following declarations:

```
char str1[] = "hello";
char *str2 = "hello";
const char *str3 = "hello";
```

State whether following statements would compile. If not explain why.

`str1 = "HELLO";` **[1 mark]**

`str2 = "HELLO";` **[1 mark]**

`str3 = "HELLO";` **[1 mark]**

(e) **[3 marks]** The following code code compiles but causes undefined behaviour. Why is this?

```
char *str = "hello";
str[0] = "H";
```

(f) **[3 marks]** Consider the following declarations.

```
void a(int **x);
void b(int (*x)[4]);
void c(int *x[]);
void d(int x[][4]);

int arr[2][4];
```

Which of `a`, `b`, `c`, or `d` could you use `arr` as an argument for? Briefly explain your answer.

Question 3. Data Structures and Memory **[20 marks]**

(a) **[6 marks]** Consider the following code:

```
typedef union
{
    int *i;
    char c[8];
} Data;

char s[][] = {{'a', 'b', 'c'}};
```

Assume that this code is running on a 32-bit system. What is the result for `sizeof` for the given statements:

`sizeof(Data)` **[1 mark]**

`sizeof(Data*)` **[1 mark]**

`sizeof(&s)` **[1 mark]**

`sizeof(s)` **[1 mark]**

`sizeof(s[0])` **[1 mark]**

`sizeof(s[0][0])` **[1 mark]**

(b) **[4 marks]** Briefly explain what a **memory leak** is and describe how it may occur when using `malloc` to allocate memory.

(c) **[2 marks]** Assume the following `malloc` is successful:

```
int *ptr = malloc(10 * sizeof(int));
```

State the possible outcomes of the following statment and discuss why a temporary variable `tmp` is used:

```
int *tmp = realloc(ptr, 100 * sizeof(int));
```

(d) **[2 marks]** Briefly explain the difference between `malloc` and `calloc`. Discuss when `malloc` should be used in preference to `calloc`.

(e) **[2 marks]** Explain why the following code would not compile.

```
int *p1 = malloc(128);
extern i = 0;

int main(void)
{
    char *p2 = malloc(128);
    return 0;
}
```

(f) **[4 marks]** What does the following print out:

```
#include <stdio.h>

static int a = 0;
int foo(int);


int main(void)
{
    static int b = 0;

    b += foo(2);
    printf("a=%d,b=%d\n", a, b);

    b += foo(3);
    printf("a=%d,b=%d\n", a, b);

    return 0;
}


int foo(int n)
{
    static int b = 0;
    int i;

    for (i = 0; i < n; ++i)
    {
        a++;
        b++;
    }

    return b;
}
```

Question 4. Projects, File I/O and Process Management **[20 marks]**

(a) **[4 marks]** Consider the following code:

```
#include <stdio.h>

char l337ify(char);


int main(int arc, char* argv[])
{
    if (arc != 3)
        return 1;

    FILE *fin, *fout;
    int c;

    if ((fin = fopen(argv[1], "r")) == NULL
    || (fout = fopen(argv[2], "w")) == NULL)
    {
        printf("Failure.\n");
        return 1;
    }

    fprintf(fout, "%d␣speak", 1337);
    while ((c = fgetc(fin)) != EOF)
    {
        fputc(l337ify(c), fout);
    }

    fclose(fin);
    fclose(fout);

    return 0;
}


char l337ify(char c)
{
    switch(c)
    {
    case 'e' : return '3';
    case 't' : return '7';
    case 'o' : return '0';
    default : return c;
    }
}
```

(Question 4 continued on next page)

(Question 4 continued)

i. **[4 marks]** Assume the program runs without errors. Given the following file as `arg[1]` what is the output to the file in `arg[2]` ?

```
Whats the good of Mercators
North Poles and Equators,
Tropics, Zones, and Meridian Lines?
```

ii. **[4 marks]** What is the difference between `"r"`, `"w"` and `"a"` as arguments for `fopen` when opening an **existing file**

(b) **[4 marks]** Discuss the difference between headers files (.h) and source files (.c) and why they are used for large projects.

(c) **[2 marks]** `exec` is a function that only returns if an error occurs. In the successful case that `exec` is called what happens to memory of the process that calls it?

(d) **[2 marks]** Briefly explain what a zombie process is and give an example of how to avoid it.

(e) **[4 marks]** Consider the following code:

```
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

int main(void)
{
    int status, id1, id2;

    id1 = fork();
    if (id1)
    {
        printf("a");
        waitpid(id1, &status, 0);
    }
    else
    {
        printf("b");
        exit(0);
    }

    id2 = fork();
    if (id2)
    {
        printf("c");
        waitpid(id2, &status, 0);
    }
    else
    {
        printf("d");
        exit(0);
    }

    printf("e\n");
    return 0;
}
```

List ALL possible outputs assuming no errors occur (Hint: draw a diagram of the processes, where they fork and where they join).

Question 5. Python Fundamentals **[20 marks]**

(a) **[4 marks]** Determine the **type AND value** of x in the following statements.

```
x = not(4 < 3)or (6 >= 9)
```

```
x = 'Hello'
x = x[2:5]
```

```
x = 10.5//4
```

```
x = {2, 3, 4}
x.add('4')
```

```
x = (1, 2, 3)
x = x + x
```

```
x = {'a':3, 'b':2, 'c':0}
x[3] = 'd'
```

(b) **[10 marks]** What does the following code print out:

i. **[2 marks]**

```
s = 'This takes the cake'
l = s.split(' ')
for w in l :
    print(w.upper())
```

ii. **[2 marks]**

```
l = list('abc')
l.extend('def')
l.append('ghi')
print(l)
```

iii. **[2 marks]**

```
l = [0, 1, 2, 3, 4, 5]
for i in range(-3, 3) :
    print(l[i])
```

(Question 5 continued on next page)

(Question 5 continued)

iv. **[2 marks]**

```
x = [1, 3, 1, 5, 2, 1, 3]
print([e**2 for e in x[1:-1]])
```

v. **[2 marks]**

```
a = {1, 2, 3, 4}
b = {3, 4, 5, 6}

print(b - a)
print(a - b)
print(a | b)
print(a & b)
```

(c) **[2 marks]** Consider the following code:

```
import os.path
print(path.abspath())
```

Explain why this code will throw a runtime error?

(d) **[2 marks]** Explain the difference between calling a function using positional arguments and calling a function using keyword arguments.

(e) **[2 marks]** What does an immutable type mean in python? Give **three** examples of an immutable type.

Question 6. Trivia **[361 marks]**

(a) **[361 marks]** 2016 and again in 2017 a computer algorithm developed by DeepMind beat professional players in an abstract strategy board game invented in ancient China more than 2,500 years ago. What is the Korean name for this board game?

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