UNIVERSITY OF SOUTHAMPTON FEEG6002C1

SEMESTER 1 EXAMINATIONS 2018-19

Advanced Computational Methods 1

DURATION 120 MINS

This paper contains 6 **Questions. Answer All Questions.**

This is a Computer-Based Exam.

All answers must be saved in the default folder "My Documents".

For all questions, template files have been provided for your use. These files can be found in My Documents*.*

A total of 100 marks are available for this paper. Marks available for answering parts of the questions are shown in brackets thus [ ].

A foreign language direct 'Word to Word' translation dictionary
(paper version ONLY) is permitted, provided it contains no notes,
additions or annotations

Version saved: 23/11/2018 10:41:39

1.

a). Write Emacs commands for the followings:
i. open or create file,
ii. save file,
iii. compile file,
iv. run the code in eshell.

b). What are the two parts of a `while` statement?

c). What are the four parts of a `for` statement?

d). How are the elements of an `array` stored in memory?

e). Show two ways to obtain the address of the first element of the array `data[ ]`.

f). If an `array` is passed to a function, what are two ways to know where the end of that `array` is?

g). To store a string of `n` characters, it is necessary to have a character array of `n+1` elements. Why is the extra element needed?

h). What keyword in the `C` programming language is used to create a structure?

Answer this question using the notepad editor to open the file `q1.txt`. You must save the file upon completion.

[20 marks]

2. Find the errors on each of the following code segments and explain how to correct them.

a).

```
#include<stdio.h>
void print_msg( void );
int main ()
{
    print_msg(“This is a message to print”);
    return 0;
}
void print_msg( void )
{
    puts(“This is a message to print”);
    return 0;
}
```

b).

```
int get_1_or_2(void)
{
    int answer=0;
    while ( answer <1 || answer >2)
    {
        printf(Enter 1 for Yes, 2 for No);
        scanf(“%f”, answer);
    }
     return answer;
}
```

c).

```
struct{
        char zodiac_sign[21];
        int month;
    } sign= “Leo”, 8;
```

TURN OVER

d).

```
    int x[3][12];
    int *ptr[12];
    ptr=x;
```

e).

```
    void *p;
    p=(float*)malloc(sizeof(float));
    *p=1.23;
```

f).

```
    union data{
        char a_word[4];
        long a_number;
    }generic_variable = {"WOW", 1000};
```

Answer this question using the notepad editor to open the file `q2.txt`. You must save the file upon completion.

[20 marks]

3. Write a function that accepts two strings. Count the number of characters in each, and return a pointer to the long string. You may extend the provided `q3.c` function to complete the solution:

```
#include<stdio.h>
main()
{

    return 0;
}
```

Answer this question using the Quincy editor to open and extend the file `q3.c`. You must save the file upon completion.

[10 marks]

4. Write a function named addarrays() that accepts two arrays that are the same size. The function should add each element in the arrays together and place the values in a third array. You may extend the provided q4.c function to complete the solution:

```
#include<stdio.h>
#define SIZE 5

main()
{
    return 0;
}
```

Answer this question using the Quincy editor to open and extend the file q4.c. You must save the file upon completion.

[10 marks]

5. In this question, the task is to write a library of functions to handle the complex numbers.

a) Define a structure complex to contain members re and im to represent real and imaginary parts of a complex number. The members re and im should be of a type double. [4 marks]

b) Define a function init(x, y) which takes as input two arguments x and y of a type double, and returns a type complex as an output. Within the function init, the input x should be assigned to the member re of the output structure, and y should be assigned to the member im of the output structure. The function init will be useful in the subsequent part of the question for initialising any variable declared as a type complex. [4 marks]

c) Define a function add(a, b) which takes two arguments a and b both of a type complex and returns a type complex as an output. Within the function add, the two complex numbers represented by a and b are added by adding their real and imaginary parts and the new complex number is returned as an output. [4 marks]

TURN OVER

d) Define a function `conjugate(a)` which takes as an input a single argument `a` of a type `complex` and returns a type `complex` as an output. The function `conjugate` is to return the complex conjugate of `a` obtained by multiplying its imaginary part by −1. [4 marks]

e) In `main()`, declare arrays `a, b, c` each having `N` elements of a type `complex`. Thus `a, b, c` represent `N`-element arrays of complex numbers. Given the array `a`, the array `b` is to be populated by complex conjugates of the elements of `a`. The array `c` is to be populated by the sums of elements of `a` and `b`, i.e. `c[i] = a[i] + b[i]` where `i` is the index of the `i`-th element of the arrays. As a test, initialise the array `a` by `N=3` complex numbers `1+1j, 2+4j, 3+9j` using your function `init`, compute the arrays `b` and `c` by using your functions `conjugate` and `add`, and you should expect the following output:

```
  1.000000+1.000000j
  2.000000+0.000000j
  2.000000+4.000000j
  4.000000+0.000000j
  3.000000+9.000000j
  6.000000+0.000000j
```

if the following code snippet (provided in `q5.c`) is populated correctly by the missing code: [4 marks]

```
#include<stdio.h>


/* Structure complex to represent a complex number */
...

/* Function init that to return a complex number
   having real part x and imaginary part y */
...

/* Function add return a sum of two complex numbers a
   and b */
...
```

```
/* Function conjugate to return a complex conjugate
   of a complex number a */
…

/* Print a complex number c in the format x + yj*/

void print(struct complex c)
{
    if(c.im >= 0.0)
        printf("%f+%fj\n", c.re, c.im);
    else
        printf("%f%fj\n", c.re, c.im);
}

int main()
{

    …

    for(i=0; i<N; i++)
    {
        print(a[i]);
        print(c[i]);
    }

    return 0;
}
```

Answer this question using the Quincy editor to open and extend the file `q5.c`. You must save the file upon completion.

[20 marks]

TURN OVER

6. Let a, b, and c represent square matrices of size n x n such that c = a*b where * implies matrix multiplication.

a) Write a function multiply(a, b, c, n) which returns no value and the arrays a, b, c representing the matrices are passed to it as pointers to type double. The input n is an integer representing the size of the matrices. The function should perform a matrix multiplication of the matrices a and b, given to the function as input, by using the direct summation $c_{ij} = \sum_{k=0}^{n-1} a_{ik} b_{kj}$ and store the result in c. In main(), allocate the arrays a, b, c by using dynamic memory allocation. [6 marks]

b) Write a function power(a, n, p), which calculates a matrix power $a^p = a \cdot a \cdots a$ by performing the p-fold matrix multiplication starting from the right-end. The function power returns no value and the square matrix a of size n x n is given to the function as a pointer to an array with all elements of type double. Both the matrix size n and the power index p are integers and assume p>0. The output after the multiplication should be stored in the array a, i.e. rewriting the initial input matrix. You can call your function multiply defined above to perform the multiplication. [10 marks]

Note that instead of using two-dimensional arrays, which requires handling pointer-to-pointer operations, matrices can be defined contiguously as one-dimensional arrays of length n*n and matrix elements ij accessed as, e.g., d[i*n + j] where i, j = 0, …, n−1. This approach was used in the code snippet shown below (provided in q6.c), which after including the missing parts, produces the following output: [4 marks]

```
|1.000000 0.000000 |
|0.000000 1.000000 |

|4.000000 4.000000 |
|4.000000 4.000000 |
```

Code in q6.c:

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

/* function multiplying two matrices */
…
/* function to compute p-th power of a matrix */
…
/* print a formatted matrix */

void print(double *a, int n)
{
    int i, j;

    for(i=0; i<n; i++) {
        printf("|");
        for(j=0; j<n; j++) {
            printf("%f ", a[i*n+j]);
        }
        printf("|\n");
    }
    printf("\n");
}

int main()
{
    …
    /* dynamic allocation of arrays*/
    …

    a[0*N+0] = 1.0;
    a[0*N+1] = -1.0;
    a[1*N+0] = 1.0;
    a[1*N+1] = 1.0;

    b[0*N+0] = 0.5;
    b[0*N+1] = 0.5;
    b[1*N+0] = -0.5;
    b[1*N+1] = 0.5;
```

TURN OVER

```
    multiply(a, b, c, N);

    print(c, N);

    a[0*N+0] = 1.0;
    a[0*N+1] = 1.0;
    a[1*N+0] = 1.0;
    a[1*N+1] = 1.0;

    power(a, N, 3);
    print(a, N);

    …
    return 0;
}
```

Answer this question using the Quincy editor to open and extend the file `q6.c`. You must save the file upon completion.

[20 marks]

END OF PAPER