Course 161/2S3, Solutions to Trinity Term paper 2000

1. $(117)_{10} = (75)_{16} = (1110101)_2$. The `HEX` pattern for this is (using 2's complement)
$$ffffff8b$$
$(181.72)_{10} = (b5.b8)_{16} = (10110101.101110000)_2$. The `HEX` pattern for this is (using IEEE)
$$4335b800$$
$(fa.3c)_{16} = (11111010.00111100)_2 = (250.234375)_{10}$. The `HEX` pattern for this is (using IEEE)
$$437a3c00$$

2.
- ```
  double fofx( double x)
  {
    return( (3*x*cos(-x)+sqrt(x))/(sqrt(x)*(2*x+1)) );
  }
  ```
- ```
  double mac( double x, int n)
  {
    int k;
    double sum=0.0;

    for(k=1; k<=n; k++)
      {
        sum += pow(-1,k-1)*pow(x,2*k-1)/(2*k-1);
      }
    return(sum);
  }
  ```
- ```
  void abs_array( int n, double x[])
  {
    int i;

    for(i=0; i<n; i++)
      {
        x[i] = fabs( x[i] );
  ```
  ```

```
        }
    }
```

3. (a) $(((x * y)/i) * j) \rightarrow 1.5$
(b) $((i + j) == (2 * j)) \rightarrow$ FALSE$(= 0)$
(c) $(((x > y) \&\& (y > j)) || (i < 3)) \rightarrow$ FALSE$(= 0)$

Describe the memory map:

byte address	variable value in memory	variable name
32		x[1]
24		x[0]
20		j-p
12		j
4		p
0		i

Trace what happens (you could just fill these values into the memory map you have already drawn:

byte address	variable value in memory	variable name
32	$\cancel{2}.0$ 2.5	x[1]
24	$\cancel{1}.0$ 2.5	x[0]
20	$\cancel{24}_{MEM}$ 32_{MEM}	j-p
12		j
4	2.5	p
0	$\cancel{0}$ $\cancel{1}$ 2	i

4. The proof that the error is $\mathcal{O}(h^3)$ is in the notes, as is the statement of the extended trapezoidal rule and the further derivation of the error for n steps of size $h = \dfrac{b-a}{n}$.

You should have derived

$$\text{error term} \;=\; \frac{1}{12}\frac{(b-a)^3}{n^2}max(f''\Big|_a^b)$$

Then for $f(x) = \dfrac{1}{x^2}$, $max(f'')_a^b = max\left(\dfrac{6}{x^4}\right)_{x=1}^{x=3} = 6$. Therefore

$$\begin{aligned}
\frac{1}{12}\frac{(3-1)^3}{n^2}6 &< 10^{-5} \\
\frac{4}{n^2} &< 10^{-5} \\
n^2 &> 4\times 10^5 \\
n &> 632.45
\end{aligned}$$

Therefore $n = 633$ sub-intervals are needed.

5. The 2 coupled equations are

$$\begin{aligned}
\frac{dx}{dt} &= z(t) \\
\frac{dz}{dt} &= -x(t)
\end{aligned}$$

Define the Euler algoritm: to solve the equation $\dfrac{dy}{dx} = f(x,y)$, the Euler algorithm is

$$\begin{aligned}
y_0 &= A \\
y_{n+1} &= y_n + hf(x_n, y_n)
\end{aligned}$$

For this problem:
t_1 ie $t = 0.1$

$$\begin{aligned}
x_1 &= x_0 + hz_0 = 1.0 \\
z_1 &= z_0 + h(-x_0) = -0.1
\end{aligned}$$

t_2 ie $t = 0.2$

$$\begin{aligned}
x_2 &= x_1 + hz_1 = 0.99 \\
z_2 &= z_1 + h(-x_1) = -0.2
\end{aligned}$$

t_3 ie $t = 0.3$

$$
\begin{aligned}
x_3 &= x_2 + hz_2 = 0.97 \\
z_3 &= z_2 + h(-x_2) = -0.299
\end{aligned}
$$

t_4 ie $t = 0.4$

$$x_4 = x_3 + hz_3 = 0.9401$$

6. The main points are

   - Call by value pass values of variables to functions. Call by reference uses pointers to pass the address of variables in `main` to the function
   - Call by value results in a local copy in the function, call by reference does not result in a local copy
   - To return a result, or change the value of a variable in the `main` program requires a `return` statement when using call by reference. This can be accomplished without a `return` statement using call by reference.

```
void swap( int *p, int *q)
{
  int tmp;

  tmp = *p;
  *p = *q;
  *q = tmp;
}
```

7. Main points

   - draw the picture as given in the notes
   - explain (briefly) the proceedure as represented in the picture
   - method arises from a Taylor series expansion ie $f(x+\delta) = f(x) + \delta f'(x) + \ldots$
   - restricting to the first 2 terms $\Rightarrow \delta = -f(x)/f'(x)$

- signal for a root at x is $\delta = 0$ and $f(x) = 0$.
- Newton Raphson algorithm is $x_{n+1} = x_n + \delta$

The proof of quadratic convergence is in the notes (referred to as a recurrence relation). The 2 roots are $x = 2.303$ and $x = -1.303$ (you should show workings for these results).

8.
```
#include <stdio.h>
#include <math.h>

typedef struct {
        double re;
        double im;
        } complex;

complex multiply_cmplx(complex a, complex b);
complex divide_cmplx(complex a, complex b)

main()
{
  complex x,y, x_times_y, x_over_y;

  printf("enter the real and imaginary parts of x\n");
  if (scanf ("%lf %lf", &x.re, &x.im) != 2)
    {
      printf("error - try again");
      exit(1);
    }

  printf("enter the real and imaginary parts of y\n");
  if (scanf ("%lf %lf", &y.re, &y.im) != 2)
    {
      printf("error - try again");
      exit(1);
    }

  x_time_y = multiply_cmplx(x,y);
```

```
  x_over_y = divide_cmplx(x,y);

  /* Print the result */

  printf("the product is %.2lf + %.2lf\nthe quotient is %.2lf + %.2lf\n ",
          x_times_y.re, x_times_y.im, x_over_y.re, x_over_y.im );
}

complex multiply_cmplx(complex a, complex b)
{
 complex product;

 product.re = (a.re*b.re - a.im*b.im);
 product.im = (a.re*b.im + a.im*b.re);

 return product;
}

complex divide_cmplx(complex a, complex b)
{
 complex quotient;

 quotient.re = (a.re*b.re+a.im*b.im)/(b.re*b.re + b.im*b.im);
 quotient.im = (a.im*b.re - a.re*b.im)/(b.re*b.re + b.im*b.im);

 return quotient;
}
```