Assignment 0. Warmup exercises Marks 15+5

Posted on 07.08.2025 @ 2:30 pm and due on 07.08.2025 @ 6:00 pm

---

1. Calculate the sum of first $N = 20$ odd numbers and factorial of $N = 8$ starting from 0. (using `do-while` or `for` loop). **[3+3]**

2. Calculate the sum of $N = 15$ terms of a GP and HP series for common difference 1.5 and common ratio 0.5 starting from $t_0 = 1.25$. Use of analytical formulae not allowed. **[3+3]**

3. Consider the following matrices,

$$\mathbf{A} = \begin{pmatrix} 2 & -3 & 1.4 \\ 2.5 & 1 & -2 \\ -0.8 & 0 & 3.1 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 0 & -1 & 1 \\ 1.5 & 0.5 & -2 \\ 3 & 0 & -2 \end{pmatrix}$$
$$\mathbf{C} = \begin{pmatrix} -2 \\ 0.5 \\ 1.5 \end{pmatrix}, \quad \mathbf{D} = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}$$

   Find $\mathbf{AB}$, $\mathbf{D} \cdot \mathbf{C}$ and $\mathbf{BC}$. **[5]**

4. ** Define *MyComplex* class / structure and calculate the sum, difference, product and modulus of $(1.3 - 2.2j)$ and $(-0.8 + 1.7j)$. **[5]**
   ** Extra 5 marks if you can do it.

Listing 1: Help with class complex : idragonlib.py

```python
import numpy as np

class MyComplex():
        def __init__(self,real,imag=0.0):
                self.r=real
                self.i=imag
        def display_cmplx(self):
                print(self.r," ",self.i,"j",sep="")
        def add_cmplx(self,c1,c2):
                self.r=c1.r+c2.r
                self.i=c1.i+c2.i
                return MyComplex(self)
        def sub_cmplx(self,c1,c2):
                self.r=c1.r-c2.r
                self.i=c1.i-c2.i
                return MyComplex(self)
        def mul_cmplx(self,c1,c2):
                self.r=c1.r*c2.r - c1.r*c2.i
                self.i=c1.i*c2.r + c1.r*c2.i
                return MyComplex(self)
        def mod_cmplx(self):
                return np.sqrt(self.r**2+self.i**2)
```

Listing 2: Help with class complex : idragoncmplx.py

```
import numpy as np
# complex math : Name -- Imagine Dragon, Roll -- 007

import numpy as np
from idragonlib import *

c1=MyComplex(0.75,1.25)
c2=MyComplex(-1.5,-2.0)
c3=MyComplex(0.0,0.0)

print("\n Two complex")
c1.display_cmplx()
c2.display_cmplx()

print("\n Adding two complex")
c3.add_cmplx(c1,c2)
c3.display_cmplx()

print("\n Subtracting two complex")
c3.sub_cmplx(c1,c2)
c3.display_cmplx()

print("\n Multiplying two complex")
c3.mul_cmplx(c1,c2)
c3.display_cmplx()

print("\n Modulus of two complex")
mod=c1.mod_cmplx()
print(f"c1 mod = {mod:.3f}")
mod=c2.mod_cmplx()
print(f"c2 mod = {mod:.3f}")
```

Run as – `python3 idragoncmplx.py`

Listing 3: Help with structure complex : idragoncmplx.c

```
// Complex math : Name -- Imagine Dragon, Roll -- 007
// compile as : gcc idragoncmplx.c -lm
// run as : ./a.out

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

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

complex setcmplx(float a,float b)
{
  complex c;
  c.re=a; c.im=b;
  return(c);
}

complex prodcmplx(complex a,complex b)
{
  complex c;
  c.re=a.re*b.re - a.im*b.im;
  c.im=a.im*b.re + a.re*b.im;
  return(c);
}

int main(int argc,char *argv[])
{
  complex z1=setcmplx(2.4,-2.7);
  complex z2=setcmplx(-0.9,1.25);
  printf("z1 = %.2f, %.2fj\n",z1.re,z1.im);
  printf("z2 = %.2f, %.2fj\n",z2.re,z2.im);

  complex z=prodcmplx(z1,z2);
  printf("z1*z2 = %.2f, %.2fj\n",z.re,z.im);

} // end of main
```