Math 1MP3 Midterm Test #2

Sample Test

1. For each of the following expressions, determine if it evaluates to **True**, **False**, or it produces an **Error**.Write your answer in the box below the expression.

In the following,

import numpy as np

- x = [0,1,2,3,4]
- a = np.array(range(5),float)
- b = np.array(x,float)
- c = np.arange(5, dtype = float)
- d = np.arange(0,5,1,dtype = float)
- e = np.array([[1,2,3],[4,5,6],[7,8,9]])

Useful tip:

np.array_equal(arr1,arr2) will return **True** if and only if arr1 and arr2 are the same, that is, **both shape and element** of two arrays are the same.

Each part of this question is worth 1 point, for a **maximum of 5 points**. No partial credit will be given and no points will be deducted for an incorrect answer.

Expression 1

```
- -4**2**-1 != 1.0/16
```

In [ ]:

```
# grouping order, precedence
```

Expression 2

```
9 - 4 ** 3 / 5 // 2 % 15 == 3.0
```

In [ ]:

```
# grouping order, precedence, floating point number
```

Expression 3

```
np.array_equal(a,d) and np.array_equal(b,c)
```

In [ ]:

```
# array creation
```

Expression 4

```
e[1,0] == x[-1]
```

In [ ]:

```
# multi-dimension array, index
```

Expression 5

```
np.array_equal(b == d ,np.array([True,True,True,True,True]))
```

In [ ]:

```
# logical arrays
```

1. For each of the following python statements determine what is printed by them and write your answer in the box below the statement. If the statement or statements result in an error, your answer should be **Error**.

In the following,

- str_1 = "I love "
- str_2 = "python programming"
- list_1 = [1,2,3]
- set_1 = {4,5,6}
- tup_1 = ("x", "y", "z")
- dict_1 = {"java": 2000, "haskell":1000, "c":{"c#":500, "c++":300}}

Each part of this question is worth 1 point, for a **maximum of 10 points**. No partial credit will be given and no points will be deducted for an incorrect answer.

Statement 1:

```
b = list_1
b[2] = 0
print(list_1+b)
```

In [ ]:

```
# aliasing, list concatenation
```

Statement 2

```
a = tup_1[1]
b = a
a = list_1[1]
print(b)
```

In [ ]:

```
# value assignment, index
```

Statement 3

```
str_3 = str_1[2:]
str_4 = str_2[7:-4]
print(str_3+str_4)
```

In [ ]:

```
# slicing, string concatenation
```

Statement 4

```
import numpy as np
a = np.array(list_1)
print(a.shape+tup_1)
```

In [ ]:

```
# array creation, tuple concatenation, shape
```

Statement 5

```
set_1.add(6)
set_1.add(7)
set_1.add(8)
print(set_1[3])
```

In [ ]:

```
# set methods, property of set
```

Statement 6

```
tup_1[0] = str(1)
```

In [ ]:

```
# mutability
```

Statement 7

```
import numpy as np
a1 = np.array(tup_1)
b1 = a1[1:3]
b1[0]="x"
print(b1)
```

In [ ]:

```
# array creation, array slicing, mutability
```

Statement 8

```
list_3 = []
for i in range(1,len(list_1)):
    list_3.append(list_1[0:3:i])
print(list_3)
```

In [ ]:

```
# range, list expansion, advanced slicing, for loop
```

Statement 9

```
for something in dict_1:
    print(something)
```

In [ ]:

```
# for loop, dictionary
```

Statement 10

```
list_1.append(set_1)
list_1.append(tup_1)
list_1.append(list_1)
list_1.append(dict_1)
print(len(list_1)//2)
print(len(list_1)%2)
print(len(list_1)/2)
```

In [ ]:

```
# list methods, arithmetics
```

1. Evaluate each of the following expressions and write the result in the box below the expression. If the expression can not be evaluated, you should write **Error** as your answer.

Each part of this question is worth 1 point, for a maximum of 3 points. No partial credit will be given and no points will be deducted for an incorrect answer.

For the following,

import numpy as np

- a = np.array([[1, 2], [3, 4], [5, 6]], float)
- b = np.array([-1, 3], float)
- c = np.array([-5, 0], float)

Expression 1

```
(a+b)[2, 0]
```

In [ ]:

```
# array mathematics, index, broadcasting
```

Expression 2

```
np.dot(a.flatten(),a.flatten())
```

In [ ]:

```
# vector mathematics, reshape
```

Expression 3

(b/c)[0,0]

In [ ]:

```
# array mathematics
```

1. For each of the functions defined below, provide a **brief** description, in the space just below the code, of what it does. Do not just describe what the individual lines of code do. Each part of this question is worth a **maximum of 2 points**

(a) In the following, **arr** is a **sorted** non-empty list with 6 integers, and **x** is an integer.

```
def fun_1 (arr,x, l=0, r=5):
    while l <= r:
        mid = l + (r - l)//2;
        if arr[mid] == x:
            return mid
        elif arr[mid] < x:
            l = mid + 1
        else:
            r = mid - 1
    return -1
```

In [ ]:

(b) In the following, **file_1** is a plain text file that is already open for reading.

```
def fun_2(file_1):
    lines = file_1.readlines()
    for i in range(1,len(lines),2):
        for word in lines[i].split():
            print(word.upper())
```

In [ ]:

(c) In the following, **arr** is an array of integers.

```
import numpy as np
def fun_3(arr, change_value, new_value):
    new_arr = arr.copy()
    change_positions = new_arr == change_value
    new_arr[change_positions] = new_value
    return new_arr
```

In [ ]:

```

```

1. **Each part of this question is worth 3 points.** Partial credits will be given as appropriate.

(a) Circle or indicate where all of the mistakes/omissions/errors in syntax and logic are in the code. In the box below the code, write a correctd version of the code.

The following function has one integer argument **n** and returns the sum of the first n terms of the harmonic series. If n is less than 1, then the function should return the value 0.0. So, sum harm(3) should return the value 1.8333333333333333, which is 1/1 + 1/2 + 1/3.

```
def sum_harm():
    if n < 1:
    return 0.0
    else:
       sum = 0
       for counter in range(1,n):
           sum += 1//counter
    return n
```

In [ ]:

```

```

(b) In the box below write code for a function **scalar_product(file1,file2)** that has two arguments: **file1** and **file2**, both both have an unknown number of lines, each line consisting of a single positive integer. The function reads a line from one file and then a line from the other file. The two integers are multiplied together and their product is added to a variable called **scalar_product** which should be initialized to zero.

Your code should stop when it detects end of file in either file that it is reading. When it stops, it should return **scalar_product**.

For example, if the sequence of integers in one file was "9 7 5 18 13 2 22 16" and "4 7 8 2" in the other file, your code would compute:

4×9 + 7×7 + 8×5 + 2×18

and thus return 161.

In [ ]: