Instructions:

1. You are allowed to bring one index card of notes (3in x 5in). Do not submit the index card with the exam.

2. This is a closed book exam. Do not confer with any other person.

3. Computers are not allowed.

4. Calculators are not allowed.

5. Show your work. Partial credit will be given. Grading will be based on correctness, clarity and neatness.

6. I suggest that you read the whole exam before beginning to work any problem. Budget your time wisely.

7. All the programs in the problems must be written in Python.

8. The duration of this exam is 75 minutes.

DO NOT BEGIN THE EXAM UNTIL INSTRUCTED TO DO SO. GOOD LUCK!

Please sign the academic integrity statement:
“On my honor, as an Aggie, I have neither given nor received unauthorized aid on this academic work. In particular, I certify that I have not received or given any assistance that is contrary to the letter or the spirit of the guidelines for this exam.”

Student Signature: ________________________________

PROCTORED EXAM ONLY

Exam Date:

Proctor Name: Proctor Signature:
1. (8 points) Circle the right answer(s) for the following statements

(a) (1 pt.) In the following conditional statement:

```python
if __name__ == "__main__":
    main()
```

`__name__` is called:

i. A method
ii. A value
iii. An argument
iv. A special variable
v. A built-in function

(b) (1 pt.) True or False. The function `random.random()` returns a random float between 0 and 100.

(c) (1 pt.) True or False. The function `random.shuffle()` chooses an element from a sequence at random.

(d) (1 pt.) True or False. The `return` command causes the execution to leave the current function immediately.

(e) (1 pt.) True or False. A function can have more than one `return` statement.

(f) (1 pt.) We can import all definitions from a module using:

i. `from module_name import all`
ii. `from module_name import definitions`
iii. `from module_name import *` (Correct answer)
iv. `from module_name import global`

(g) (1 pt.) We can rename an imported module using:

i. `import module_name as new_name` (Correct answer)
ii. `import module_name to new_name`
iii. `rename module_name to new_name`
iv. `import module_name from new_name`

(h) (1 pt.) A _____ variable is declared outside of the functions, and can be accessed by any functions in the program

i. local
ii. built-in
iii. scope
iv. global
2. (18 points) Consider the following Python program.

*Hint: one of the outputs generates an error.*

```python
a = {3, 9, 5, 8}   # Line A
b = [[3, 6, 2, 8], [7, 5, [6, 9]], [8, 1], [1, 5]]  # Line B
c = ["pixel", 2, "ram", 8, "core"]
d = [21, 15, 18, 19]

print(b[2])      # Line C
print(b[1][2])   # Line D
print(b[1][2][0])# Line E
print(len(b))    # Line F
print(c[c[1]])   # Line G
print([num % 2 for num in d]) # Line H
print(a[2])      # Line I
```

(a) (2 pts.) What is the datatype of the variable on Line A? Set
(b) (2 pts.) What is the datatype of the variable on Line B? List
(c) (2 pts.) What is the output of Line C? [8, 1]
(d) (2 pts.) What is the output of Line D? [6, 9]
(e) (2 pts.) What is the output of Line E? 6
(f) (2 pts.) What is the output of Line F? 4
(g) (2 pts.) What is the output of Line G? ram
(h) (2 pts.) What is the output of Line H? [1, 1, 0, 1]
(i) (2 pts.) What is the output of Line I? Error: A set is unordered object does not support indexing
3. (9 pts) What is the output of the following program?

```python
specs = ["core", "ram", "cpu"]
result = ""
for choice in specs:
    for letter in choice:
        if letter in "mocu":
            result += letter

print(result)  # Line A
print(choice)  # Line B
print(letter)  # Line C
```

(a) (3 pts.) What is the output of Line A? comcu
(b) (3 pts.) What is the output of Line B? cpu
(c) (3 pts.) What is the output of Line C? u

4. (12 pts)

(a) (3 pts) Write three differences between lists and sets in Python.

List:
- Ordered elements
- Duplicates elements are possible
- Indexable

Set:
- Unordered elements
- Unique elements
- Elements are immutable;

(b) (3 pts) Which function would you use to get all the keys from a Python dictionary? 
.stdout(keys())

(c) (3 pts) Which function would you use to get all the items from a Python dictionary?
.stdout(items())

(d) (3 pts) Which function would you use to pick a random item from a Python list?
.stdout(choice())
5. (9 pts) Write the Python line(s) of code that:

(a) (3 pts.) Write the code to add a number $n$ at the end of a list $A$.

```
A = []
n = 2
A.append(n)
print(A)
```

(b) (3 pts.) Write the code to select a group of 3 names randomly from the list $\text{students} = \text{["nate", "sam", "anna", "tim", "jane"]}$.

```
import random
students = ["nate", "sam", "anna", "tim", "jane"]
print(random.sample(students, 3))
```

(c) (3 pts.) Write the code to create a list $A$ of 20 random elements between 0 and 50.

```
import random
A = []
for n in range(20):
    A.append(random.randint(0, 50))
```

---

6. (12 pts) What is the output of the following program?

```
specs1 = {"core", "ram", "cpu"}
specs2 = {"ram", "bios", "cpu", "os"}
union = specs1.union(specs2)
intersection = specs1.intersection(specs2)
diff1 = specs1.difference(specs2)
diff2 = specs1.symmetric_difference(specs2)
print(union)  # Line A
print(intersection)  # Line B
print(diff1)  # Line C
print(diff2)  # Line D
print({"bios"}.issubset(specs1))  # Line E
```

(a) (2.5 pts.) What is the output of Line A? {'cpu', 'bios', 'core', 'ram', 'os'}

(b) (2.5 pts.) What is the output of Line B? {'cpu', 'ram'}

(c) (2.5 pts.) What is the output of Line C? {'core'}

(d) (2.5 pts.) What is the output of Line D? {'os', 'bios', 'core'}

(e) (2 pts.) What is the output of Line E? False
7. (15 pts) What is the output of the following program?

```python
ports = {"HDMI": 2, "USB": 4, "DVI": 5, "VGA": 1}
print (ports.keys())       # Line A
print (ports.values())     # Line B
print (ports["HDMI"])     # Line C
print (ports.get("USB"))   # Line D
rports = {}
for connector, number in ports.items():
    rports[number] = connector
print (rports)             # Line E
interface = {}
for number in sorted(rports):
    interface[number] = rports[number]
print (interface)          # Line F
```

(a) (2.5 pts.) What is the output of Line A? `dict_keys(['HDMI', 'USB', 'DVI', 'VGA'])`
(b) (2.5 pts.) What is the output of Line B? `dict_values([2, 4, 5, 1])`
(c) (2.5 pts.) What is the output of Line C? 2
(d) (2.5 pts.) What is the output of Line D? 4
(e) (2.5 pts.) What is the output of Line E? `{2: 'HDMI', 4: 'USB', 5: 'DVI', 1: 'VGA'}`
(f) (2.5 pts.) What is the output of Line F? `{1: 'VGA', 2: 'HDMI', 4: 'USB', 5: 'DVI'}`
8. (10 pts) An isogram is a word that has no repeating letters (consecutive or nonconsecutive). Create a function `is_isogram` that takes a word as input and returns either True or False depending on whether or not the word is an isogram. The function is not case sensitive.

Sample output:

```
def is_isogram(word):
    unique_letters = len(set(word.lower()))
    letters = len(word)
    return letters == unique_letters

while True:
    word = input("Enter a word: ")
    print(f"Isogram: {is_isogram(word)}\n")
```

```
Enter a word: Jumpy
Isogram: True

Enter a word: computer
Isogram: True

Enter a word: America
Isogram: False

Enter a word: Halloween
Isogram: False
```
9. (7 pts) Write a Python program that prompts the user for a number \( n \) and calculates the factorial of that number \( n \). The factorial of a positive integer \( n \) called \( n! \) is the product of all positive integers less than or equal to \( n \). e.g. \( 5! = 5 \times 4 \times 3 \times 2 \times 1 \).

\( n \) should be non-negative. If \( n \) is negative, the program should print an error message.

**Sample output:**

```
Enter a number: 5
Factorial: 120
```

```
n = int(input("Enter a number: "))
factorial = 1
if n < 0:
    print("Enter a positive number")
elif n == 0:
    print(1)
else:
    for i in range(1, n + 1):
        factorial = factorial * i
    print(f"Factorial: {factorial}"
```

10. (Extra Credit: 2 points) Write a Python program that prompts the user for a list of numbers. Create a function that takes in the list and returns a list of the accumulating sum. Print the accumulating sum.

**Sample output:**

```
Enter numbers: 1 2 3 4 5
Accumulating sum: 1 3 6 10 15

Enter numbers: 3 1 5 8 2
Accumulating sum: 3 4 9 17 19
```

```
def accumulating_list(lst):
    if len(lst) == 0:
        return []
    else:
        accumulator = []
        mysum = 0
        for item in lst:
            mysum += item
            accumulator.append(mysum)
return accumulator
```

def accumulating_list(lst):
    if len(lst) == 0:
        return []
else:
    accumulator = []
    mysum = 0
    for item in lst:
        mysum += item
        accumulator.append(mysum)
return accumulator
```

def accumulating_list(lst):
    if len(lst) == 0:
        return []
else:
    accumulator = []
    mysum = 0
    for item in lst:
        mysum += item
        accumulator.append(mysum)
return accumulator
```

```
n = int(input("Enter a number: "))
factorial = 1
if n < 0:
    print("Enter a positive number")
elif n == 0:
    print(1)
else:
    for i in range(1, n + 1):
        factorial = factorial * i
    print(f"Factorial: {factorial}"
```

```
def accumulating_list(lst):
    if len(lst) == 0:
        return []
    else:
        accumulator = []
        mysum = 0
        for item in lst:
            mysum += item
            accumulator.append(mysum)
        return accumulator
```

```
n = int(input("Enter a number: "))
factorial = 1
if n < 0:
    print("Enter a positive number")
elif n == 0:
    print(1)
else:
    for i in range(1, n + 1):
        factorial = factorial * i
    print(f"Factorial: {factorial}"
```

```
def accumulating_list(lst):
    if len(lst) == 0:
        return []
    else:
        accumulator = []
        mysum = 0
        for item in lst:
            mysum += item
            accumulator.append(mysum)
        return accumulator
```

```
n = int(input("Enter a number: "))
factorial = 1
if n < 0:
    print("Enter a positive number")
elif n == 0:
    print(1)
else:
    for i in range(1, n + 1):
        factorial = factorial * i
    print(f"Factorial: {factorial}"
```

```
def accumulating_list(lst):
    if len(lst) == 0:
        return []
    else:
        accumulator = []
        mysum = 0
        for item in lst:
            mysum += item
            accumulator.append(mysum)
        return accumulator
```

```
n = int(input("Enter a number: "))
factorial = 1
if n < 0:
    print("Enter a positive number")
elif n == 0:
    print(1)
else:
    for i in range(1, n + 1):
        factorial = factorial * i
    print(f"Factorial: {factorial}"
```

```
def accumulating_list(lst):
    if len(lst) == 0:
        return []
    else:
        accumulator = []
        mysum = 0
        for item in lst:
            mysum += item
            accumulator.append(mysum)
        return accumulator
```

```
n = int(input("Enter a number: "))
factorial = 1
if n < 0:
    print("Enter a positive number")
elif n == 0:
    print(1)
else:
    for i in range(1, n + 1):
        factorial = factorial * i
    print(f"Factorial: {factorial}"
```

```
def accumulating_list(lst):
    if len(lst) == 0:
        return []
    else:
        accumulator = []
        mysum = 0
        for item in lst:
            mysum += item
            accumulator.append(mysum)
        return accumulator
```

```
n = int(input("Enter a number: "))
factorial = 1
if n < 0:
    print("Enter a positive number")
elif n == 0:
    print(1)
else:
    for i in range(1, n + 1):
        factorial = factorial * i
    print(f"Factorial: {factorial}"
```