Part 1 – Implement the Monte Carlo Inventory Algorithm given on page 210 in Python

In **Step Six** call the following function to implement the demand model

```python
from random import *  # access random library

def getDemand(x):
    # returns uniform value between lo and hi
    # for x a random float between 0 and 1
    hi = 1800
    lo = 1200
    return lo + x * (hi - lo)
```

This demand model assumes that demand is uniformly distributed between a low (1200 gallons) and a high (1800) value. 1500 is the average daily demand.

**Hints, Notes and Requirements**

1. Begin your code by inputting the following parameters:

   ```python
def main():
    Q = eval(input("Enter delivery quantity (gallons): "))
    T = eval(input("Enter time between deliveries in days: "))
    d = eval(input("Enter delivery cost in dollars: "))
    s = eval(input("Enter storage costs per gallon per day: "))
    N = eval(input("Enter number of days to run simulation: "))
```

2. Between Steps 8 and 9 it’s useful to display the daily inventory. I inserted the following print statement where i was my loop counter for the for loop (Steps 5 – 9)

   ```python
   print("day:",i,"Inventory:",round(I),"gallons")
   ```

3. Steps 5 – 9 are contained within a For loop. Step 10 follows the For loop.

4. Use the build-in function `round(n,k)` to round n to k digits

5. Don’t forget to import the random library
Part 2: Implement the Harbor System simulation Algorithm found on page 216 in Python.

Begin with the following code which allows user to input number of ships (n) then allocates lists of size n needed to implement the algorithm

```
from random import *

def main():
    n = eval(input("Enter number of ships: "))

    # Allocate lists
    between = [0]*n
    unload = [0]*n
    arrive = [0]*n
    finish = [0]*n
    idle = [0]*n
    wait = [0]*n
    harbor = [0]*n
    start = [0]*n

    # Generate random scenario inputs
    for i in range(n):
        between[i] = randrange(15, 46)
        unload[i] = randrange(15,46)

    # fixed values from page 214
    # between = [20, 30, 15, 120, 25]
    # unload = [55, 45, 60, 75, 80]

    Note that 2 lines have been commented out above that allows the code to run using the ship scenario given on page 214. This is useful for testing purposes. Un-comment when testing; commend out when not.

2. Use the following output code

```
print()
print("Average time per ship in harbor: ",HARTIME)
print("Maximum time of ship in harbor: ",MAXHAR)
print("Average waiting time per ship before unloading: ",WAITIME)
print("Maximum waiting time of a ship: ",MAXWAIT)
print("Finish Time: ",finish[n-1])
print("Total Idle Time: ",IDLETIME)
IDLETIMEPCNT = IDLETIME/finish[n-1] * 100
print("Percent total harbor idle time: ",round(IDLETIMEPCNT,2),"%")
```
3. It also might be useful to print out at Step 10 when a ship finishes

   print("Ship", i+1, "finishes at", finish[i])

Of course don’t forget to include the same output at step 3 when the first ship finishes.

   print("\nShip 1 finishes at", finish[0])

4. Finally note that Python uses zero based indexing while the algorithm assume one based indexing.

   Hand in

   1. Source code for a program must begin with the standard header comment block which lists your name, the file name, the date and a brief description of what your program does.

   2. Hand in hardcopy of source code and one or more sample runs. Open a NotePad document and copy your sources code and your sample runs to it. Sample runs for a program must appear immediately after the source code.

   3. Store copies of your source code files in your individual Q:\ drive folder.