NYU – Tandon School of Engineering
Brooklyn, NY 11201

CS-UY 1114 Intro to Programming & Problem Solving

Second Midterm Exam – 22 November 2022

Name: SAHIL SINGH

NetID (first part of NYU email): SKS9379

Exam Seat Number: G 4

- Duration: 1 hour and 15 minutes.
- **DO NOT WRITE ON THE BACK OF ANY PAGE!**
- **Do not separate any page (9 pages).**
- **Please do not use pencil**, if you must, write darkly, these pages will be scanned.
- **If you believe there is an error, please write ERROR for your answer.**
- If you write an answer other than in the space provided, please indicate, in the space provided, where we can find that answer.
- This is a closed book exam; no calculators are allowed.
- You can expect that the user inputs the appropriate values (int/float/etc where required).
- Comments are not required.
- Anyone found cheating on this exam will receive a zero for the exam.
- Anyone who is found writing after time has been called will receive a zero for this exam.
- Do not open this test booklet until you are instructed to do so.
- If you have a question, please ask the proctor of the exam!
- You may use ONLY the following Python constructs and function:

`(all math operators)`	`if,elif,else`	`for`
`(all conditional operators)`	`while`	`ord`
`math.* (all in math module)`	`int`	`chr`
`random.* (all in random module)`	`print`	`input`
`str.* (all string methods)`	`len`	`str`
`list.* (all list methods)`	`del`	`def,return`
`dict.* (all dictionary methods)`	`in`	`is`

- For reference, the point distribution for each question is below:

Q1) (12pts; 3 pts each)
Q2) (16pts; 4 pts each)
Q3) (20pts; 4 pts each)
Q4) (20pts)
Q5) (32pts)

Name Sahil Net ID: SKS9779

1) (12pts; 3 pts each; **Circle one answer per segment**)
Given these 3 lists, for each of the following code segments, indicate whether it will generate an error or not. Consider each code segment as independent.

```
list_1 = [1,2,3]
list_2 = ['a','b','c']
list_3 = [list_1, ["do", "re", "me"], list_2]
```

Code segment	ERROR or NO ERROR?	
`list_1[0] = list_2[0:2]` `List_1[1][2] = 'x'`	(ERROR)	NO ERROR
`list_2.append(list_1.popitem())`	ERROR	(NO ERROR)
`value = list_3[1][0][:2]` `print(value)`	(ERROR)	NO ERROR

`my_pairs = []`
`for i in range(len(list_3[1])):`
`  my_pairs.append((list_1[i], list_2[i]))`
`my_pairs[1][0] = 100`
	ERROR	(NO ERROR)

2) (16 pts; 4 pts each)
Given this code segment, for each of the following print statements, show what is printed. Write "ERROR" if an error would occur. If nothing is printed then write "NO OUTPUT".

```
x = -200
def kiwi(x):
  if x < 100:
      return "small"
  else:
      return "large"

def grapes(x):
  print("grapes:", x)
  return x + 7

def watermelon(y):
  y = grapes(y)
  print("watermelon:", y)
```

Statement	Output
`print(grapes(10))`	grapes: 10 17
`print(watermelon(2))`	grapes: 2 9 watermelon: 9
`print(kiwi(120))`	"large"
`print(kiwi(grapes(2)))`	grapes: 2 9 Small

Name Sahil Net ID: SKS9774

3) (20 pts; 4 pts each)
Given the following dictionary, evaluate these statements and show what prints treating the code as *one continuous program.* Write "ERROR" if an error would occur. If nothing is printed then write "NO OUTPUT".

```
st_ca_dict = {
      1: {'PA': ['Harrisburg', 49528]},
      2: {'CO': ['Denver', 715522]} ,
      3: {'GA': ['Atlanta', 498715]},
      4: {'AR': ['Little Rock', 202591]},
      5: {'MD': ['Annapolis' , 38394]},
      6: {'NY': ['Albany', 97856]},
      7: {'TX': ['Austin', 961855]}
}
```

Statement	Output

`st_ca_dict.pop(6)`
`my_list = list(st_ca_dict.keys())`
`print(my_list)`
	'NY': ['Albany', 97856]
	['PA', 'CO', 'GA', 'AR', 'MD', 'TX']

`print(st_ca_dict[4]['AR'])`	['Little Rock', 202591]
`print(st_ca_dict[4]['AR'][1:])`	202591
`st_ca_dict[6] = st_ca_dict.pop(7)` `print(len(st_ca_dict))`	5

`st_ca_dict.clear()`
`st_ca_dict[8] = {'WA': ['Olympia', 46478]}`
`print(st_ca_dict)`
	{'WA': ['Olympia', 46478]}

Name SAHILSINGH Net ID: SKS9379

4) (20 pts) **If the following file were to be executed, what would the output be?**

```
def function_one(one, two):
      print(one, end='')
      print(function_three(), end='')
      print(two, end='')

def function_two(one='a'):
      print(one, end='')
      function_four(one, 'c')

def function_three():
      return chr(ord('e') + 1)

def function_four(one, two):
      print(one + two, end='')
      return 'f', 'g'

def main():
      print('h', end='')
      function_one('i', 'j')
      print('k', end='')
      function_two()
      print('m', end='')
      value = function_four('n', 'o')
      print(value[1], end='')

print('s', end='')
main()
print('r', end='')
```

hifgK

Output:

ERROR
As function two can accept one parameter but in the main(), function two is being called with no parameter

Name SANIL SINGH Net ID: SKS9979

5) *Programming Problem (32 pts)*

Worst Bakers in America is an American cooking competition television series that airs on Food Network. In this show the country's worst bakers are recruited into two teams (a red team and a blue team) and pitted against each other for a grueling six weeks to see what they can accomplish. Part of the competition is to have the two teams face off to craft the most delicious high-end cake. Both teams must use the same recipe. However, each team has their own separate food pantry.

We would like to write a computer program to help them determine the bakeable cake recipes that both teams can make given the following options and available ingredients in their food pantries:

`RED_PANTRY` **is a dictionary mapping ingredients to counts of those ingredients available in the food pantry of the red team.**

```
RED_PANTRY = {'egg': 12, 'cup of sugar': 4, 'cup of flour': 4, 'chocolate': 10}
```

`BLUE_PANTRY` **is a dictionary mapping ingredients to counts of those ingredients available in food pantry of the blue team:**

```
BLUE_PANTRY = {'egg': 100, 'cup of sugar': 4, 'cup of flour': 2, 'chocolate': 5}
```

`CAKE_RECIPES` **is a nested dictionary mapping cake names to the ingredients and required amount:**

```
CAKE_RECIPES = {
    'sugary cake' : {'cup of sugar': 4, 'cup of flour': 4},
    'simple cake' : {'egg': 1 , 'cup of sugar': 2, 'chocolate': 1 },
    'milky cake'  : {'egg': 10 , 'cup of sugar': 2, 'milk': 100 },
    'eggy cake'   : {'egg': 100, 'cup of sugar': 1, 'milk': 4},
    'choco cake'  : {'egg': 2, 'cup of sugar': 2, 'chocolate': 4}
}
```

A cake recipe is considered *bakeable* if *both* teams (red and blue) have at least enough of every ingredient required by the recipe.

1. **Write the function** `fetch_bakeable()` (*sig. dict, dict, dict –> list[str]*) **that accepts the two pantries and the cake recipes as arguments and returns a list of all the *bakeable* cake names that both teams can make using the available ingredients from the two pantries.**

```
fetch_bakeable(RED_PANTRY, BLUE_PANTRY , CAKE_RECIPES)
```

Output:

```
['simple cake', 'choco cake']
```

2. **Write a** `main()` (*sig. None –> None*) **function that correctly invokes the function** `fetch_bakeable()`**. You can assume that** `RED_PANTRY`, `BLUE_PANTRY` , `CAKE_RECIPES` **are all already defined at the top of your file.**

Name SAHIL SINGH Net ID: SKS9379

```
# solution for problem 5
def fetch_bakeable(RED_PANTRY, BLUE_PANTRY, CAKE_RECIPES):
    list=[]
    for i in RED_PANTRY.key():
        list.append(i)

    for j in BLUE_PANTRY.key():
        list.append(j)

    list1=[]
    for i in range(len(CAKE_RECIPES)-1):
        cake=CAKE_RECIPES[i].value()
        lol=cake.key()
        if lol in list:
            list1.append(CAKE_RECIPES[i])

    return list1



get main():
    bake=fetch_bakeable(RED_PANTRY, BLUE_PANTRY, CAKE_RECIPES)
    print(bake)

main()
```

Name _____ Net ID: _____

```
# solution for problem 5
```

Name _____ Net ID: _____

```
# (Additional Space)
```

Name _____ Net ID: _____

```
# (Additional Space)
```