Assignment 7

Assignment not submitted

Due date: 2020-03-18, 23:59 IST.

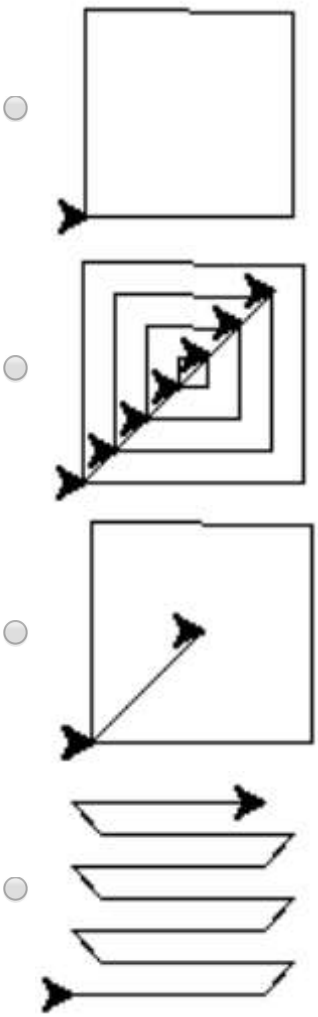
Predict the output ***1 point***

```
l=[[1,2,3],[4,5,6],[7,8,9]]
f=1
for j in range(3):
  if(f==1):
    for i in range(3):
      print(l[i][j],end=" ")
      f=0
  if(f==0):
    for i in range(2,-1,-1):
      print(l[i][j],end=" ")
      f=1
```

- 7 4 1 1 4 7 2 5 8 8 5 2 3 6 9 9 6 3
- 1 4 7 2 5 8 3 6 9
- 1 4 7 8 5 2 3 6 9
- 1 4 7 7 4 1 2 5 8 8 5 2 3 6 9 9 6 3

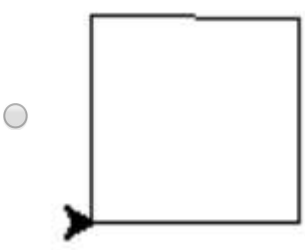
Predict the output of the calling function func1() for a given square matrix mx of dimension 70 × 70. ***1 point***

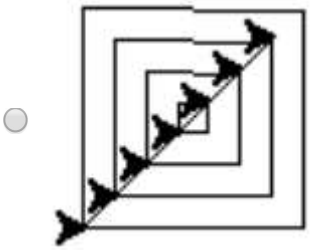
```
def func(mx,i):
  tur = turtle.Turtle()
  tur.setpos(i,i)

  for ind in range(i,n-i):
    tur.goto(i,ind)
  for ind in range(i+1,n-i):
    tur.goto(i,n-1-i)
  for ind in range(n-2-i,i,-1):
    tur.goto(n-1-i,ind)
  for ind in range(n-i-1,i,-1):
    tur.goto(ind,i)

def func1(mx):
  n=len(mx)
  i=0
  while(i<=n-1):
    func(mx,i)
    i=i+10
```

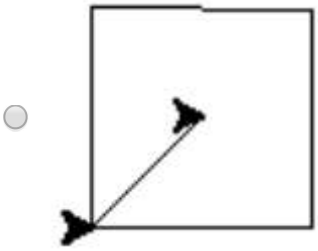


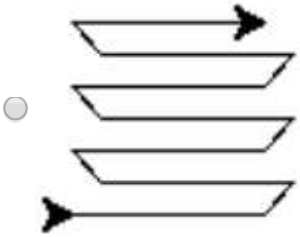
Predict the output of the calling function func() for a given square matrix mx of dimension 70 × 70. ***1 point***

```
def func(mx):
  func1(mx,0)
  tur = turtle.Turtle()
  tur.setpos(0,0)
  if((len(mx))%2==1):
    turtle.goto(int(len(mx)/2),int(len(mx)/2))
  else:
    second=int(len(mx)/2)
    turtle.goto(second-1,second-1)
    turtle.goto(second-1,second)
    turtle.goto(second,second-1)
    turtle.goto(second,second)

```









Which of the following libraries is required to work with Google maps in Python? ***1 point***

- gplot
- googleplot
- gmplot
- none of these

Which of the following codes represent a correct version of a board game where the user has to move from block 1 to block 100? ***1 point***

The game initialises only when the user gets a 1 or 6 on the dice and ends once he reaches 100 or gets a number which makes him reach beyond 100 (i.e. the player wins if he is at 99 and gets a 4).

```
import random
def play(psn):
  r = random.randint(1,6)
  if(psn==0):
    if(r==1 or r==6):
      psn=1
    else:
      psn=psn+r
  print("Position=",psn)
  if(psn>=100):
    print("You won")
    return
  play(psn)
position=0
print("Position=",position)
play(position)
```

```
import random

def play(psn):

  r = random.randint(1,6)
  print("Dice rolled:",r)
  if(psn==0):
    if(r==1 or r==6):
      psn=1
  else:
    psn=psn+r
  print("Position=",psn)
  if(psn>=100):
    print("You won")
    return
  play(psn)


position=0
print("Position=",position)
play(position)
```

```
import random

def play(psn):

  r = random.randint(1,6)
  print("Dice rolled:",r)
  input()
  if(psn==0):
    if(r==1 or r==6):
      psn=1
  else:
    psn=psn+r
  print("Position=",psn)
  if(psn>=100):
    print("You won")

  play(psn)


position=0
print("Position=",position)
play(position)
```

```
import random

def play(psn):

    print("Dice rolled:",2)
    if(psn==0):
        psn=1
    else:
      psn=psn+2
    print("Position=",psn)
    if(psn>=100):
      print("You won")

    play(psn)
position=0
print("Position=",position)
play(position)
```

Imagine a single player snakes and ladders game. The code below represents ***1 point***

```
import random

def play(psn):
  snake_begin=-1
  snake_end=-1
  while(snake_begin <= snake_end):
    snake_begin=random.randint(1,99)
    snake_end=random.randint(1,99)
  print("Snake from",snake_begin,"to",snake_end)
  r = random.randint(1,6)
  print("Dice rolled:",r)
  if(psn==0):
    if(r==1 or r==6):
      psn=1
  else:
    psn=psn+r
  print("Position=",psn)
  input()
  if(psn==snake_begin):
    print("Bitten by snake")
    psn=snake_end
  if(psn>=100):
    print("You won")
    return
  play(psn)

position=0
print("Position=",position)
play(position)
```

- A snakes and ladders game with one snake whose position remains constant while the player is playing. The position also remains the same during any subsequent plays (i.e. the game board does not change while you sleep and play again the next day).
- A snakes and ladders game with one snake whose position remains constant while the player is playing.However, the position can change during any subsequent plays (i.e. the game board might change while you sleep and play again the next day).
- A snakes and ladders game with one snake where the snake can change its position during the game and also during any subsequent plays (a board game where the snakes keep moving). Further, the snake can bite you any number of times.
- A snakes and ladders game with one snake where the snake can change its position during the game and also during any subsequent plays (a board game where the snake keeps moving). Further, the snake can bite you only ones when you play.

Imagine a single player snakes and ladders game. The code below represents ***1 point***

```
import random

def play(psn,flag):
  snake_begin=-1
  snake_end=-1
  while(snake_begin <= snake_end):
    snake_begin=random.randint(1,99)
    snake_end=random.randint(1,99)
  print("Snake from",snake_begin,"to",snake_end)
  r = random.randint(1,6)
  print("Dice rolled:",r)
  if(psn==0):
    if(r==1 or r==6):
      psn=1
  else:
    psn=psn+r
  print("Position=",psn)
  #input()
  if(psn==snake_begin and flag==0):
    print("Bitten by snake")
    psn=snake_end
    flag=1
  if(psn>=100):
    print("You won")
    return
  play(psn,flag)

position=0
print("Position=",position)
play(position,0)
```

- A snakes and ladders game with one snake whose position remains constant while the player is playing. The position also remains the same during any subsequent plays (i.e. the game board does not change while you sleep and play again the next day).
- A snakes and ladders game with one snake whose position remains constant while the player is playing. However, the position can change during any subsequent plays (i.e. the game board might change while you sleep and play again the next day).
- A snakes and ladders game with one snake where the snake can change its position during the game and also during any subsequent plays (a board game where the snakes keep moving). Further, the snake can bite you any number of times.

A snakes and ladders game with one snake where the snake can change its position during the game and also during any subsequent plays (a board game where the snake keeps moving). Further, the snake can bite you only ones when you play.

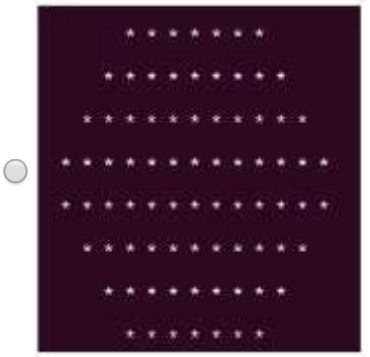
Assuming that the play1() function implements the recursive play of snakes and ladders with the prespecified position of the snake and the ladder as shown in the code below, which of the ambiguities in the options can result in the code? Kindly assume a typical snakes and ladders game. ***1 point***

```
import random
def play(psn):
  snake_begin=-1
  snake_end=-1
  while(snake_begin <= snake_end):
    snake_begin=random.randint(1,99)
    snake_end=random.randint(1,99)
  ladder_begin=-1
  ladder_end=-1
  while(ladder_end <= ladder_begin):
    ladder_begin=random.randint(1,99)
    ladder_end=random.randint(1,99)
  play1(psn,snake_begin,snake_end,ladder_begin,ladder_end)
```

- snake_begin=snake_end
- ladder_begin=ladder_end
- ladder_begin=snake_begin
- ladder_end=snake_end

What is the output of the following code? ***1 point***

```
import random
def play(psn):
  snake_begin=-1
  snake_end=-1
  while(snake_begin <= snake_end):
    snake_begin=random.randint(1,99)
    snake_end=random.randint(1,99)
  ladder_begin=-1
  ladder_end=-1
  while(ladder_end <= ladder_begin):
    ladder_begin=random.randint(1,99)
    ladder_end=random.randint(1,99)
  play1(psn,snake_begin,snake_end,ladder_begin,ladder_end)
```



What is the output of the following code? ***1 point***

```
def func():
    print()
    c=10
    i=3
    while(i<=6):
        j=0
        while(j<=20):
            if(j>=10-i and j<=10+i):
                print('*',end=" ")
            else:
                print(' ',end=" ")
            j=j+1
        print('\n')
        i=i+1
    i=6
    while(i>=3):
        j=0
        while(j<=20):
            if(j>=10-i and j<=10+i):
                print('*',end=" ")
            else:
                print(' ',end=" ")
            j=j+1
        print('\n')
        i=i-1
func()
```

Submit Answers number of times before the due date. The final submission will be considered for grading.