UNIVERSITY OF TORONTO
UNIVERSITY TRIBUNAL
TRIAL DIVISION

IN THE MATTER charges of academic dishonesty made on February 22, 2010;

AND IN THE MATTER of the *University of Toronto Code of Behaviour on Academic Matters, 1995*;

AND IN THE MATTER of the *University of Toronto Act, 1971*, S.O. 1971, c. 56, as amended S.O. 1978, c. 88;

BETWEEN:

THE UNIVERSITY OF TORONTO

– AND –

X   : Z

Hearing Date: May 7, 2010

Members of the Panel:
Ms. Julie Hannaford, Barrister and Solicitor, Chair
Professor Louis Florence, Department of Management, Faculty Panel Member
Mr. Sadek Ali, Student Panel Member

Appearances:
Mr. Robert Centa, Assistant Discipline Counsel, Paliare Roland Barristers
Ms. Betty-Ann Campbell, Law Clerk, Paliare Roland Barristers
Mr. Gregory Ko, Articling Student, Paliare Roland Barristers
Mr. Kante Easley, Instructor for CSC104H1F: The How and Why of Computing
Mr. Yan Zhang, Student in CSC104H1F: The How and Why of Computing

In Attendance:
Ms. Tamara Jones, Academic Integrity Officer (former), Office of Academic Integrity
Mr. Christopher Lang, Coordinator, Appeals, Discipline and Faculty Grievances

Not in Attendance:
Mr. X   Z   :, the Student

1. On May 7, 2010, the University Tribunal convened a hearing to address charges against Mr. Z — namely that he had obtained unauthorized assistance in connection with certain coursework in a course called "CSC 104H: The Why and How of Computing". At the hearing, the Tribunal heard evidence about the efforts made to contact Mr. Z, and the numerous modes of communication engaged to give Mr. Z notice of the charges, and of the pending proceedings, and as well, to give Mr. Z notice of the substance of the evidence that would be tendered about both the notice given to him and the evidence that the University was intending to put forth in regard to the charges. The Tribunal determined that based on all the evidence about the extensive communications to Mr. Z, that he should be regarded as having been "served" and having had "notice" of the proceedings. Having had no response whatsoever from Mr. Z, the Tribunal determined to proceed in his absence. In particular, the Tribunal found that the University had provided reasonable notice to Mr. Z of the hearing and that he had declined to participate in it.

FINDINGS OF FACT BY THE TRIBUNAL

2. At the hearing, the University tendered a number of documents, collectively marked as Exhibit 1. In addition, the University called the evidence of Professor Kante Easley, and the evidence of Mr. Y Z.

3. Professor Easley was the instructor in the 2008 Fall Session of CSC 104H. Mr. X Z was enrolled in Professor Easley's class.

4. Part of the requirements for the course included an assignment that was due on December 3, 2008. This assignment was worth ten percent of the final grade in the course.

5. Mr. Z submitted the required assignment on December 3, 2008. It became evident to Professor Easley that the assignment contained answers identical to those submitted by Mr. Y Z who was another student in the class. Mr. Y Z was enrolled in the Fall Session of CSC 104H, as well. Mr. Y Z admitted that he and Mr. X Z collaborated in formulating the answers to the questions in assignment number 4.

6. It is difficult to describe the nature of the collaboration unless the assignment and the answers of X Z and Y Z 's submission are viewed together.

7. The compared documents are attached to this set of reasons, along with the assignment.

8. The Tribunal was assisted in its analysis and the nature of the collaboration and the similarities of between the two assignments, by the colour-coded descriptions of the similarities in the two submissions. It was clear from these documents that there were substantial and remarkable similarities between the two assignments.

9. On the basis of the evidence provided by the University, including the documents that were marked as Exhibit 1, the Tribunal was unanimous in its finding of guilt.

PENALTY

1. Mr. Z committed previous offences while at the University. With respect to the offences in which he was involved, these reasons include a copy of the University Tribunal reasons in the matter of charges of academic dishonesty made on January 5, 2009, against him. In essence, the Tribunal determined, again in the absence of Mr. Z that his papers submitted in the course AST251 consisted almost entirely of plagiarized material. The Tribunal found that the plagiarism in that case was "flagrant and serious".

2. It is notable that Mr. Z did not attend at the hearing that occurred on September 10, 2009, notwithstanding that extensive efforts were made to bring the charges and the Notice of Hearing to Mr. Z 's attention.

3. Ms. Betty Ann Campbell, a law clerk at Palaire Roland, testified that "she had personally attended at the examination facility on McCaul Street, in Toronto, on March 20, 2009. [Mr. Z was writing an exam on that date]. Mr. Z identified himself to her. She then personally served the student with the Notice of Hearing, the charges and the University's disclosure brief".

4. Notwithstanding this personal service, Mr. Z did not attend at that hearing.

5. On hearing the evidence called by the University, and on hearing submissions by discipline counsel, Mr. Z was found guilty of the academic offence of unauthorized assistance in connection with any form of academic work to section B.I.1(b) of the *Code of Behaviour on Academic Matters*. The sanctions imposed were: (a) he shall receive a final grade of zero in the course CSC 104H: The Why

and How of Computing; (b) he be suspended from the University from May 4, 2010, until May 3, 2015; (c) the sanction shall be recorded on his academic record and transcript from the date of the Order until he graduates from the University.

COSTS

6.   That did not end the matter. The Tribunal asked the University to make submissions on costs – namely, whether costs should be awarded, and, if so, the appropriate amount of costs, and the terms and conditions related to the payment of costs. These reasons determine the issue of costs.

7.   The request to the University for submissions on costs emanated from the panel, and not from the University.

8.   The Tribunal was of the view that it was appropriate to canvass the question of costs, after hearing of the extensive work that was undertaken to contact Mr. Z   , and after considering the cost to the University of giving such notice.

9.   The University of Toronto *Code of Behaviour on Academic Matters* ("Code") confers jurisdiction on the Tribunal to award costs of proceedings at trial. Section C.II.(a)17(b) provides:

> Where it is considered to be warranted by the circumstances, the chair of a hearing may in his or her discretion award costs of any proceedings at trial, and may make orders as to the party or parties to and by whom and the amounts and manner in which such costs are to be paid.

10.   There is, therefore, jurisdiction for the panel, or indeed, the chair alone [in his or her discretion] to award costs, and there is a broad discretion about the quantum,

terms and conditions related to such costs.

11. That jurisdiction exists to award costs does not mean that every case is one where costs ought to be awarded. Rather, the award of costs should relate to the "circumstances" that would logically call for an award of costs. Such circumstances should relate to whether a party has been obliged to undertake an extraordinary expenditure of time and money in order to obtain a just resolution of the particular case.

12. The Code requires adherence to a standard of fairness, honesty, and integrity, - from both the teacher and the student, and the Code provides for penalties for failing to abide by the standards prescribed.

13. At the same time, the Code requires that the University adhere to the standards of fairness in proceedings connected with charges of academic misconduct under the Code. Those standards require that notice be given, that evidence be tendered, and that findings of fact, and conclusions about the charges, all proceed in an environment of utmost fairness and transparency. Obviously, there is a cost to adhering to the dictates of fairness. The fact that the University is put to this cost and must proceed in accordance with the highest standards of procedural fairness – should not be part of the consideration of whether costs are ordered. Teachers and students alike are entitled to the entire spectrum of procedural fairness without fear of costs sanctions, and without any expectation that simply adhering to the requirements of procedural fairness will garner any reward.

14. It is when a party confounds the process of delivering a fair and transparent

process for the determination of a charge that consideration of costs sanctions should arise. This case is one of such instances.

15. As set out above, this was not the first time that Mr. Z̵ had been the subject of charges and proceedings before the Tribunal. On September 10, 2009, the University Tribunal heard the case in Mr. Z̵'s absence and found that he had committed the academic offence of plagiarism. In that case, the University had personally served Mr. Z with the notice of hearing and charges. Mr. Z declined to participate in any way in the proceeding. The University went to great lengths to ensure Mr. Z was served with notice of those proceedings.

16. The University also went to great lengths to ensure Mr. Z̵ was served with these instant proceedings.

17. And so, the University Tribunal has now convened twice to hear allegations of academic misconduct against Mr. Z . He did not attend either hearing.

18. If Mr. Z did not wish to participate in the hearing, all he needed to do was to provide that information to the University by way of letter or email. Had he done so, the matter could have proceeded much more expeditiously and without some of the expenses incurred.

19. While the Tribunal has a jurisdiction to award costs, the jurisdiction has only been exercised infrequently. In 2006, in a case called University of Toronto and

P₁ D₁ (Case 441; April 7, 2007), the University Tribunal made a costs order against a student. In this case, it was not the substance of the charges,

or the penalty, that founded the costs order. Rather, it related to the fact that the student had caused the University to engage significant expense in attempting to serve the student with notice. Mr. D was charged with falsifying and circulating two academic records in November 2003 and January 2004. The Tribunal found Mr. D to have committed the offences and recommended that he be expelled from the University. The Tribunal, at the request of the University, made an award of costs against the student. In particular, the Tribunal ordered the student to pay $1,660.96 to the University in compensation for disbursements incurred by the University to locate and serve the student, who had evaded service.

20. Mr. Z caused the University to expend significant costs to locate him, and the University has expended this time and effort twice. These circumstances militate in favour of a costs award. The appropriate amount of costs should be determined on a principled basis.

21. There are two "scales" of costs – one relates to what is often called "partial indemnity"; the other relates to "substantial indemnity" – often called "solicitor/client" costs. Partial indemnity costs allow for part of the costs of the university to be recovered. Substantial indemnity costs allow for a very significant proportion of costs to be recovered.

22. It is arguable that a student who avoids service, or does not respond to notice properly given, should be prepared to have the full weight of a costs sanction befall him or her. Such conduct is insensitive to the cost of maintaining an

environment of fairness and a high standard of procedural fairness. No doubt, such a significant cost sanction might well have a deterrence effect in the University community. That type of sanction does not, however, accord with the principle of proportionality. Nor does it take into account that the cost to the University in this case (of serving Mr. Z   and presenting evidence at his hearing) would have been incurred by the University regardless of whether he responded or not.

23.   The principles of fairness and proportionality suggest that a "punitive" award of costs (substantial or even full indemnity) should be reserved for cases where there has been egregious and extraordinary behaviour. It might be reserved for cases where the party has engaged in conduct that has displayed a profound disrespect for the process and by extension, the Code. This accords with the principles set out in Mr. K. (Case 1999/00; April 20, 1999), in which the Discipline Appeals Board adopted the common law principle that an award of costs on a substantial indemnity basis (formerly called a solicitor-client basis) is only appropriate where there has been reprehensible, scandalous or outrageous conduct by a party.

24.   The Provost has submitted that there are two different ways to calculate the appropriate amount of costs to be paid by Mr. Z   First, the Tribunal could order Mr. Z   to pay the partial indemnity costs based on the entire costs of the hearing. Calculated on a partial indemnity basis, those costs total $4,677.75. Alternatively, the Tribunal could order Mr. Z   to pay only those incremental costs directly associated with his failure to participate in the hearing process. It is not possible to calculate precisely the amount incurred in that respect. However,

the Provost's best estimate is that the University incurred $1,200.15 in legal fees (calculated on a partial indemnity basis) directly due to Mr. Z 's failure to participate in the hearing process. The Provost requests that the Tribunal order Mr. Z to pay costs to the University fixed in the amount of $1,200.15. The Provost submits that this amount is reasonable, and does not include any of the University's costs associated with organizing the Tribunal hearing, or the University's internal costs associated with the attempts to contact Mr. Z at the divisional level.

25. The Provost's submissions fairly comb out the costs associated with providing Mr. Z with the notice and procedural fairness to which he is ordinarily entitled. It is the costs associated with the failure to participate in the hearing that should be borne by Mr. Z . Mr. Z is ordered to pay costs to the University, fixed at $1200.15, with interest on those costs from the date that this order is made, to the date of payment, calculated at 2% annually, not compounded. Mr. Z is required to pay these costs before he may register at the University.

Dated at Toronto, this _15_ day of November, 2010

_____
Ms. Julie Hannaford, Co-Chair

ASSIGNMENT CSC104

CSC104 ASSIGNMENT 4

Due at 7:00p.m. on December 3rd

1. INSTRUCTIONS

In this assignment you will be developing functions that could be used in the development of an online card game. For each function you should submit:

- A python function with the exact name described below. The functions should perform as specified and be placed in a file called a4.py.
- Sufficient comments to explain the purpose of your function, and the algorithm you used.

2. USEFUL TIPS

2.1. Nesting Lists. We have learned to create and manipulate single lists. For this exercise, we will work with lists of lists. We can nest any number of lists within a list. For example:

```
>>> M = [ [0,1, 2, 3], [4, 5, 6, 7], [8, 9, 10, 11] ]
>>> M[0]
[0, 1, 2, 3]
>>> M[1]
[4, 5, 6, 7]
>>> M[1][0]
4
>>> M[1][1]
5
```

You need to try this out in Wing. For example, try the following:

```
a = list()
a.append(list()) # create a new list nested in the list a
a.append(list()) # create another new list nested in the list a
a.append(list()) # create another new list nested in the list a
print len(a) # print the length of a (it should be 3)
# append items to individual lists within a
a[0].append("first")
a[0].append("second")
a[1].append("third")
a[2].append("fourth")
a[2].append("fifth")
a[2].append("sixth")
#verify length and contents of the lists
print a
print a[0]
print len(a[2])
print a[2]
print len(a[1])
print a[1]
```

2.2. Random Selection. This assignment will depend on randomly generated hands of cards. There is a module which can be used to shuffle a list of items (such as cards). The module name is random. Below is an example of python code that uses the random module to shuffle a list.

```
import random # put this line at the top of your a4.py file
```

```
>>>test_list = [5,6,7,8,9]
>>>random.shuffle(test_list)
>>>print test_list()
[6, 5, 9, 7, 8]
```

You can remove the last element of a list with the pop() function:

```
>>>test_list = [5,6,7,8,9]
>>>x = test_list.pop()
>>>print x
9
>>>print test_list()
[5, 6, 7, 8]
```

2.3. Cards and Hands. For the purposes of our program, we will represent our deck of cards as:

```
deck = [ \
'2c' , '3c' , '4c' , '5c' , '6c' , '7c' , '8c' , '9c' , 'Tc' , 'Jc' , 'Qc' , 'Kc', 'Ac', \
'2h' , '3h' , '4h' , '5h' , '6h' , '7h' , '8h' , '9h' , 'Th' , 'Jh' , 'Qh' , 'Kh' , 'Ah', \
'2s' , '3s' , '4s' , '5s' , '6s' , '7s' , '8s' , '9s' , 'Ts' , 'Js' , 'Qs' , 'Ks' , 'As', \
'2d' , '3d' , '4d' , '5d' , '6d' , '7d' , '8d' , '9d' , 'Td' , 'Jd' , 'Qd' , 'Kd' , 'Ad' ]
# note that we use the backslash \ to continue a line to the next line
```

Each quoted string in the list represents a card in the deck. The first character in each card is the rank, and the second character is the suit. An A represents an ace and can be considered low or high (i.e., a 1 or after a king), as required.

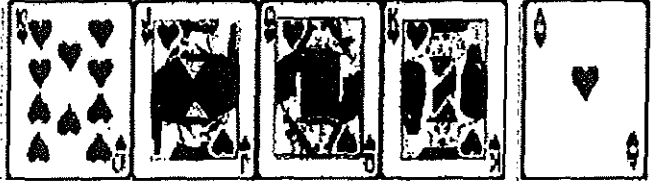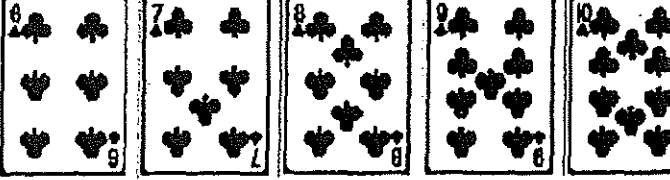Each hand that we will deal for our poker game will consist of five cards. We will keep each hand in a separate list. For example:

```
hand1 = ['3d', '4s', '4h', '7c', 'Jc']
```
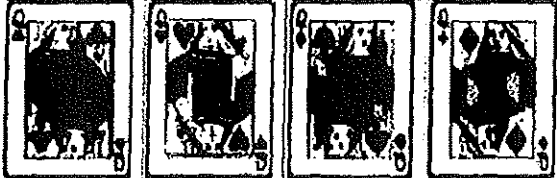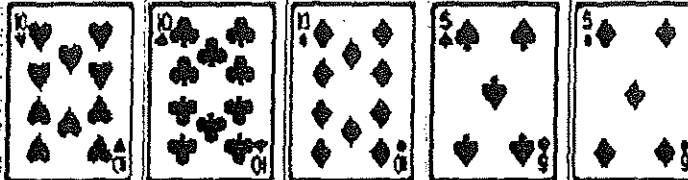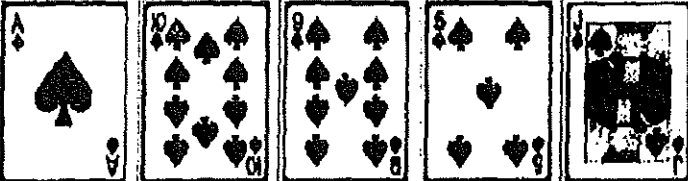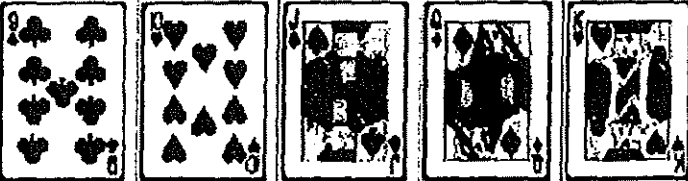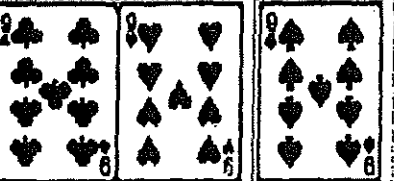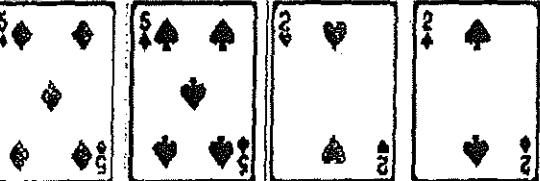
In situations where we deal more than one hand, we will nest all of the lists which contain a hand in a single list. For example:

```
hands = [['3d', '4s', '4h', '7c', 'Jc'], ['Ac', 'Ah', 'As', 'Ad', 'Ks']]
```

In this example we have nested two hands, for our game, we could nest up to ten hands.

2.4. Hands.

Royal Flush This is the best poker hand you can have and consists of a 10, Jack, Queen, King and Ace of the same suit. Ex: Tc, Jc, Qc, Kc, Ac	
Straight Flush Five cards, all of the same suit that are in sequence. Ex: 6c, 7c, 8c, 9c, Tc	

Four-Of-A-Kind
Four cards of equal rank.
Ex: Qc, Qh, Qd, Qs

Full House
Three cards of equal rank, and two different cards
of the equal rank. (Three of a kind and a pair).
Ex: Th, Tc, Td, 5s, 5d

Flush
Any five cards of the same suit.
Ex: As, Ts, 9s, 5s. 2s

Straight
Five cards of mixed suits, in sequence.
Ex: 9c, Th, Js, Qd, Kh

Three-Of-A-Kind
Any three cards of equal rank.
Ex: 9c, 9h, 9s

Two Pair
Two cards of equal rank and a different two cards
of equal rank.
Ex: 5d, 5s, 2h, 2s

Pair
Two cards of equal rank.
Ex: Js, Jh

3. THE FUNCTIONS

You must create the following functions in your program file a4.py. They must perform exactly as described. That is, if you are asked to return a list, return a list, if you are asked to return a nested list, return a nested list. Use the cards provided in the starter code. You may create additional lists for your own purposes, but the cards must be represented as in the deck description provided. Do NOT print out the returned values within the functions that you create. To test the functions, you may print out the values.

Function	Specification
deal_hands(num_hands)	Accept the number of hands to be dealt (num_hands), and use the random module to deal that number of hands. Each hand must contain five cards. No card across all of the hands may be duplicated. For example, if one hand has the three of spades (3s), that hand cannot contain another three of spades, and no other hand that is dealt may contain the three of spades. Return the hands dealt as lists nested within a single list, where each list is a hand. If there is only one hand, it must be nested as well.
royal_flush(hand)	Accept a hand (a list containing five cards). Return True if the hand is a royal flush. Otherwise, return False.
straight_flush(hand)	Accept a hand. Return True if the hand is a straight flush. Otherwise, return False.
four_of_a_kind(hand)	Accept a hand. Return True if the hand is a four of a kind. Otherwise, return False.
full_house(hand)	Accept a hand. Return True if the hand is a full house. Otherwise, return False.
flush(hand)	Accept a hand. Return True if the hand is a flush. Otherwise, return False.
straight(hand)	Accept a hand. Return True if the hand is a straight. Otherwise, return False.
three_of_a_kind(hand)	Accept a hand. Return True if the hand is a three of a kind. Otherwise, return False.
two_pair(hand)	Accept a hand. Return True if the hand is a two pair. Otherwise, return False.
pair(hand)	Accept a hand. Return True if the hand is a pair. Otherwise, return False.

COMPARED DOCUMENTS

```
greywolf:/u/csc104h/fall/submit% cat c7z         A4/a4.py
import random


deck = ['2c','3c','4c','5c','6c','7c','8c','9c','Tc','Jc','Qc','Kc','Ac',\
        '2h','3h','4h','5h','6h','7h','8h','9h','Th','Jh','Qh','Kh','Ah',\
        '2s','3s','4s','5s','6s','7s','8s','9s','Ts','Js','Qs','Ks','As',\
        '2d','3d','4d','5d','6d','7d','8d','9d','Td','Jd','Qd','Kd','Ad',]




def deal_hands(num_hands):


    random.shuffle(deck)
    # the random numbers extracted from the 'deck' were shuffled

    hand_list = []
    #pick up the required cards from deck
    for i in range(num_hands):
        hand_list.append(deck[5 * i : 5 * (i + 1)])

    return hand_list




def royal_flush(hand):

    royal_flush = False
    value_list = []

    for c in hand:
        value_list.append(c[0])

    if check_same_suit(hand):
        value_list.sort()

        #right values contains in the list shows the list
        if value_list == ['A', 'J', 'K', 'Q', 'T']:
            royal_flush = True

    return royal_flush
```

```python
#return True if hand is a full house and False otherwise
def full_house(hand):

    numerized_list = card_numerizer(hand)
    pair = False
    three_of_a_kind = False

    #this part checks if there are at exactly three cards of equal value in
    #the hand, it is similar to the three_of_a_kind function
    for i in range(2, 15):
        counter = 0

        for c in numerized_list:
            if i == c:
                counter += 1

        if counter == 3:
            three_of_a_kind = True

    #this part checks if there are at exactly two cards of equal value in
    #the hand, it is similar to the three_of_a_kind function
    for i in range(2, 15):
        counter = 0

        for c in numerized_list:
            if i == c:
                counter += 1

        if counter == 2:
            pair = True

    if pair and three_of_a_kind:
        return True
    else:
        return False

#return True if hand is a flush and False otherwise
def flush(hand):

    return check_same_suit(hand)

#return True if hand is a straight and False otherwise
def straight(hand):

    straight = True
```

```python
def two_pair(hand):

    two_pair = False

    if four_of_a_kind(hand):
        two_pair = True

    elif pair_counter(hand) == 2:
        two_pair = True

    return two_pair


#return True if hand is a pair,otherwise return False


def pair(hand):

    pair_count = pair_counter(hand)

    if pair_count > 0:
        return True
    else:
        return False
```

greywolf:/u/csc104h/fall/submit% cat c8z!   5/A4/a4.py

```python
import random
def deal_hands(num_hands):

    deck = ['2c','3c','4c','5c','6c','7c','8c','9c','Tc','Jc','Qc','Kc','Ac',\
        '2h','3h','4h','5h','6h','7h','8h','9h','Th','Jh','Qh','Kh','Ah',\
        '2s','3s','4s','5s','6s','7s','8s','9s','Ts','Js','Qs','Ks','As',\
        '2d','3d','4d','5d','6d','7d','8d','9d','Td','Jd','Qd','Kd','Ad',]
    #this contains 52 cards
    random.shuffle(deck)

    hand_list = []
    #this part creates the required amount of hands by slicing the shuffled deck
    for i in range(num_hands):
        hand_list.append(deck[5 * i : 5 * (i + 1)])

    return hand_list

def royal_flush(hand):

    royal_flush = False
    value_list = []

    for a in hand:
        value_list.append(a[0])

    if check_same_suit(hand):
        value_list.sort()

        #this is how the order of the list
        if value_list == ['A', 'J', 'K', 'Q', 'T']:
            royal_flush = True

    return royal_flush


def straight_flush(hand):

    result= False

    if check_same_suit(hand):
        result = True

        a_list = card_numerizer(hand)

        #this part is prepared for the following
```

```python
        #the hand, it is similar to the three_of_a_kind function
    for i in range(2, 15):
        counter = 0

        for x in numerized_list:
            if i == x:
                counter += 1

        if counter == 3:
            three_of_a_kind = True

    #this part checks if there are at exactly two cards have equal value in
    #the hand, it is similar to the three_of_a_kind function
    for i in range(2, 15):
        counter = 0

        for x in numerized_list:
            if i == x:
                counter += 1

        if counter == 2:
            pair = True

    if pair and three_of_a_kind:
        return True
    else:
        return False

#return True if hand is a flush and otherwise False
def flush(hand):

    return check_same_suit(hand)

def straight(hand):

    straight = True
    numerized_list = card_numerizer(hand)

    numerized_list.sort()
    #this part checks if it is a straight hand
    for i in range(4):
        if numerized_list[4] != 14:
            if numerized_list[i] != numerized_list[i + 1] - 1:
                straight = False

        elif numerized_list[4] == 14:
```

```python
if pair_count > 0:
    return True
else:
    return False
```