Intro to Coding MOOC Challenge

Team Orange

Poppy Tillott, Pamela Seale, Radhalakshmi Sankar 27th June 2023

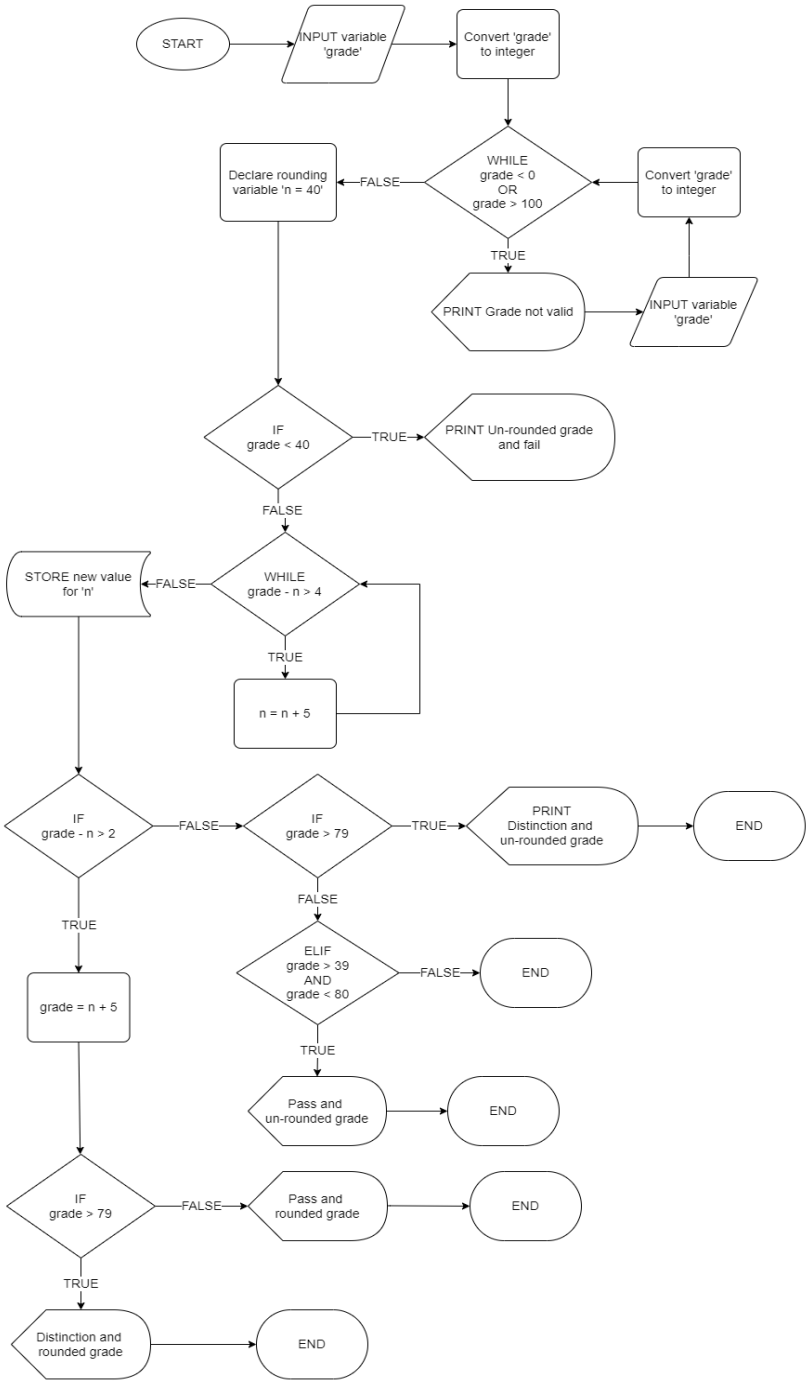
Challenges completed

- CHALLENGE 1 - Round up the final grade
- CHALLENGE 2 - Sorting an array
- CHALLENGE 5 - Hide the credit card digits

Task 1 :

ROUND UP THE FINAL GRADE

Flowchart



Task 1 :

ROUND UP THE FINAL GRADE

Pseudocode

This algorithm will determine whether an inputted grade should be rounded up and will output a rounded or un-rounded percentage grade with its classification (fail, pass or distinction).

```
grade = INPUT: "What's the grade? "
Convert grade to an integer

Check for a valid grade.
WHILE:
  grade is less than 0 OR greater than 100:
      PRINT "Invalid grade, grade must be between 0 and 100."
      grade = INPUT "What's the grade? "

Check grade is below passing threshold.
Set the initial comparison value, 40, which is the passing threshold.
IF:
 grade is less than passing threshold 40:
    PRINT "Final grade and fail"
    END
ELSE:
  IF:
    grade above passing threshold 40:
    WHILE:
      the difference between the grade and comparison value is greater than 4:
              comparison value = comparison value + 5

Manual rounding, only rounding up when the difference between the grade and the next multiple of 5 is less than 3.
IF:
 the difference between the grade and comparison value is greater than 2:
      grade = comparison value + 5
      IF:
        grade is equal to or above distinction threshold, 80:
          PRINT "Final rounded grade and distinction"
          END
      ELSE:
        grade is within pass threshold:
          PRINT "Final rounded grade and pass"
          END
ELSE:
  no rounding required
  IF:
   grade is equal to or above distinction threshold, 80:
      PRINT "Final grade and distinction"
      END
ELSE IF:
  grade is equal to or above fail threshold,40 and below distinction threshold, 80:
    PRINT "Final grade and pass"
    END
```

Task 1 :

ROUND UP THE FINAL GRADE

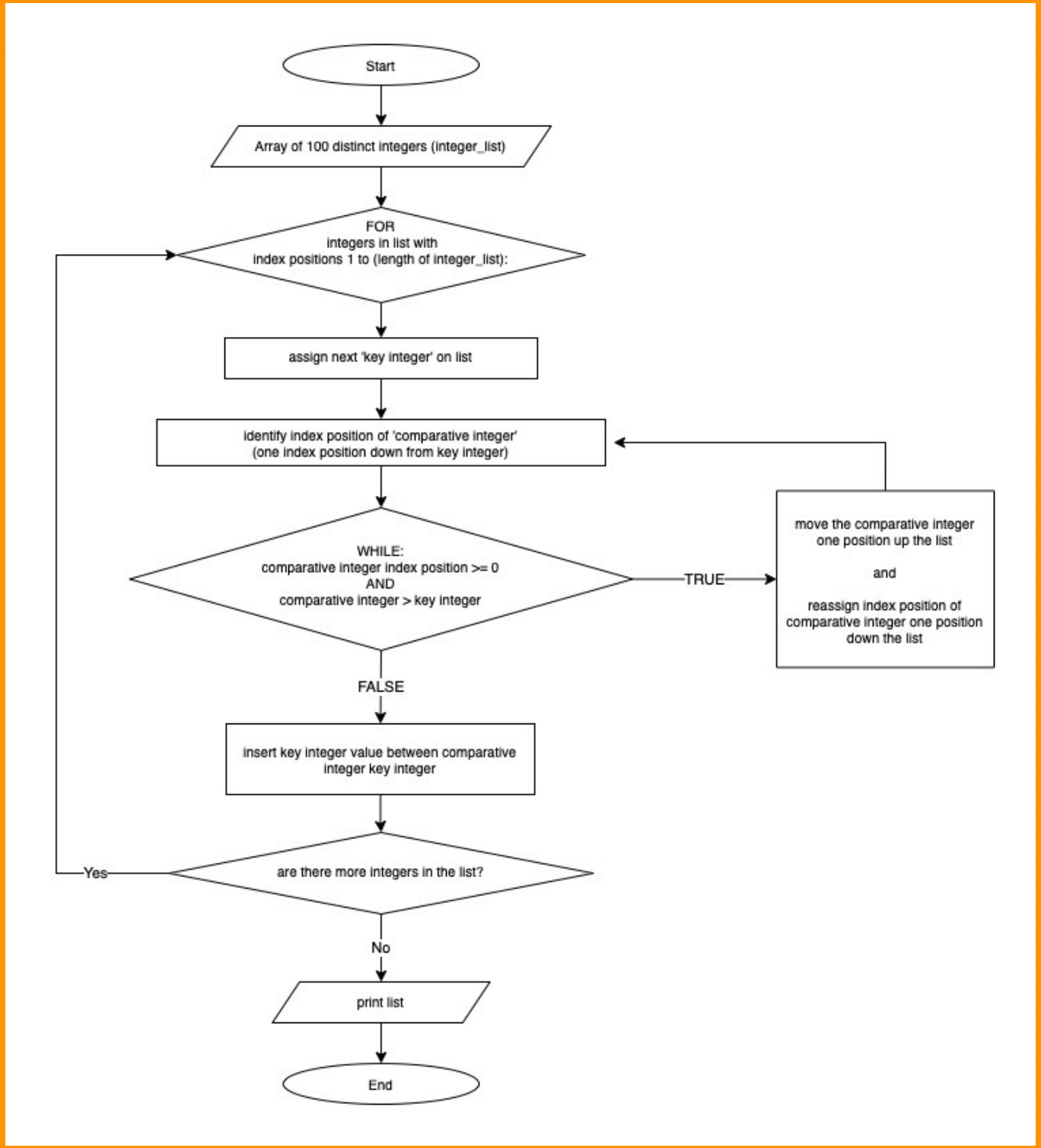
Python

```
#Obtain original grade from user
grade = int(input("What is the grade? "))
#Set initial rounding value
n = 40

#Check grade is within boundaries and return error message if not and request a valid grade
while grade < 0 or grade > 100:
    print("\n The grade boundaries are between 0 and 100. Please enter a valid grade")
    grade = int(input("What is the grade? "))



#Checking the grade is below the passing threshold and if so, notifying the user of a fail
if grade < 40:
    print(f"\n The final grade is {grade} %. The candidate has failed to pass the test.")
#Checking the difference between the grade and orginal rounding value, and changing that
#value in multiples of 5 until it's under 4
else:
    while grade - n > 4:
                n=n+5
#Manual rounding. Using the appropriate value of n to round up when n is a multiple of 5
#and returning the final grade and result category to user.
if grade - n > 2:
    grade = n+5
    if grade > 79:
        print(f"The candidates final score is {grade}% and they achieved a distinction.")
    else:
        print(f"The candidates final score is {grade}% and they passed the test.")
else:
    if grade > 79:
        print(f"The candidates final score is {grade}% and they achieved a distinction.")
    elif grade > 39 and grade <80:
        print(f"The candidates final score is {grade}% and they passed the test.")
```

Task 2 :

SORTING AN ARRAY

Flowchart



Task 2 :

SORTING AN ARRAY

Pseudocode

This algorithm will sort an array of 100 distinct integers from smallest to largest. It will do this by comparing each integer - starting with the second on the list - with each of the integers to it's left in turn, inserting it back into the list when it finds its correct position.

```
FUNCTION sort_integer_list:
        FOR 2nd integer to the last in list:
           Assign integer as 'key' integer
           Assign the integer to the left of the key integer as 'comparable' integer

        WHILE it is TRUE that the comparative integer is greater than the key integer AND there are still integers to left of the key integer, then
              change the value of the integer to the right of the comparable integer with the value of the comparable
              reassign the integer one place to its left as the new 'comparable'

        OTHERWISE:
              overwrite the integer to the right of the comparable with the key integer value

        return integer list


FUNCTION main:
          PRINT "The original list of random unique integers is: [list] "
          CALL sort_integer_list
          PRINT "The sorted list of random unique integers is: [list] "
```

Task 2 :

SORTING AN ARRAY

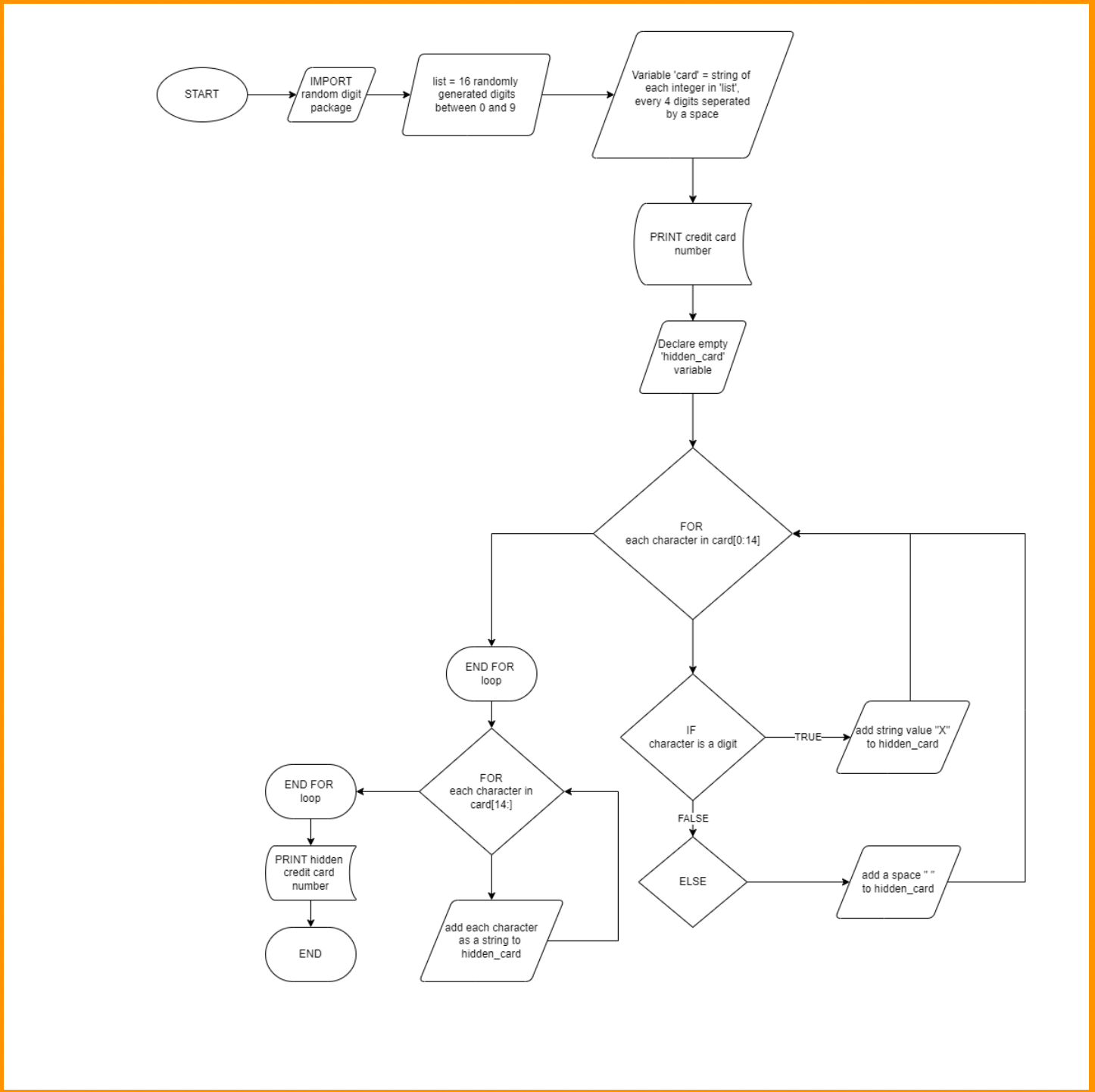
Python

```python
import random # imports random module

def create_random_set():
    # sets are unordered and do not contain duplicates so this a useful way to
    #generate the integers we will use in the task
    # creates 'set' of 100 unique random integers
    random_set = set(random.sample(range(-1000, 1000), 100)) # creates 'set' of 100 unique random integers.
    return random_set

def set_to_list(random_set):
    # converts the 'set' to a list 'int_list'
    integer_list = list(random_set)
    return integer_list

def sort(integer_list):
    # FOR all integers in list from the second to the last:
    for i in range(1, len(integer_list)):
    # assigns the variable 'k' (key integer) to each integer in the FOR loop
        k = integer_list[i]
    # assigns the variable 'c' (comparative integer) as the integer to the left of 'k'
        c = i - 1

        # WHILE LOOP repeating this function effectively shifts all integers which are larger than the
        #key integer (i.e. out of order) one position to the right.
        # WHILE there remains integers on the list AND the comparative integer is greater than the key integer:
        while c >= 0 and integer_list[c] > k:
            # change the value of the key integer to the value of the comparable integer
            integer_list[c + 1] = integer_list[c]
             # and reassign variable 'c' to the integer to the left of 'c'
            c -= 1
        # reassign variable 'k' to the integer to the right of 'c'
        integer_list[c + 1] = k
    return integer_list

def main():
    create_random_set()
    random_set = create_random_set()
#     print(random_set)
    set_to_list(random_set)
    integer_list = set_to_list(random_set)
    print(f"The original list of random unique integers is: \n{integer_list}\n")
    sort(integer_list)
    print(f"The sorted list of random unique integers is: \n{integer_list}\n")

main()
```

Task 5 :

HIDE THE CREDIT CARD DIGITS

Flowchart



Task 5 :

HIDE THE CREDIT CARD DIGITS

Pseudocode

This pseudocode will produce a randomly generated credit card number and then hide its first 12 digits.

```
IMPORT random digits package

Create a list of 16 randomly generated integers valued between 0 and 9

Declare the 'card' variable - a string of each integer from the list, every 4 digits seperated by a space

PRINT credit card number

Declare the empty 'hidden_card' variable

FOR
  each character in card from index position 0 to 13:
     IF
       the character is a digit:
            add string value "X" to hidden_card variable
     ELSE:
       add a space " " to hidden_card
FOR
  each character in card from index position 14 to the last index value:
      add string character to hidden_card

PRINT hidden credit card number
```

Task 5 :

HIDE THE CREDIT CARD DIGITS

Python

```
#Import package for random digits
import random

#Create a list of 16 random integers between 0 and 9
list = [random.randint(0,9) for i in range(16)]

#Declare variable card - string of each integer in list, every four seperated by a space
card = str(list[0]) + str(list[1]) + str(list[2]) + str(list[3]) + " " + str(list[4]) + str(list[5]) \
       + str(list[6]) + str(list[7]) + " " + str(list[8]) + str(list[9]) + str(list[10]) + str(list[11]) \
       + " " + str(list[12]) + str(list[13]) + str(list[14]) + str(list[15])
#Output randomly produced credit card number
print(f"Credit card number: {card}")
#Declare variable hidden_card
hidden_card = ""
#Loop through each element in card up to index 13
for i in card[:14]:
    #Check if a digit
    if i.isdigit():
        #If so, add string value X to hidden_card
        hidden_card += "X"
    else:
        #Otherwise add a space
        hidden_card += " "
#Loop through last space and four card digits, adding them to hidden_card
for i in card[14:]:
    hidden_card += str(i)
#Output the credit card number with its hidden values
print(f"Hidden credit card number: {hidden_card}")
```