PRE- RELEASE VOTING SYSTEM PROGRAM

2021

Carole Youssef

11.7 4966

CONTENTS PAGE:

TABLE IDENTIFIER:

Variable name	Data type	Declaration	Description
tutor_group	string	DECLARE tutor_group: STRING	variable used to store the name of the tutor group of the class
num_students	integer	DECLARE num_students: INTEGER	variable used to store the number of students in the tutor group
num_candidates	integer	DECLARE num_candidates: INTEGER	variable used to store the number of candidates in the election
candidate_names	string	DECLARE candidate_names: STRING	variable used to store the names of the candidates in the election
names	1D array	DECLARE names : ARRAY [1 TO candidate_names] OF STRING	1 dimensional array used to store all the names of the candidates
vote_count	1D array	DECLARE vote_count : ARRAY [1 TO num_candidates] OF INTEGER	1 dimensional array used to count the votes each candidate receives
student_unique	1D array	DECLARE student_unique: ARRAY [1 TO unique_num] OF INTEGER	variable used to store all the unique ID numbers of the students
unique_num	integer	DECLARE unique_num: INTEGER	variable used to store the unique ID number of each student in the class
student_choice	string	DECLARE student_choice: STRING	variable used to ask the user whether they want to vote or abstain
student_abstention	integer	DECLARE student_abstention : INTEGER	variable used to store and count the number of abstentions in the class
candidate_votes	integer	DECLARE candidate_votes: INTEGER	variable used to store and count the number of votes for each candidate in the election
total_votes	integer	DECLARE total_votes : INTEGER	variable used to find the total number of votes in the class
percentage_votes	1D array	DECLARE percentage_votes: ARRAY [1 TO num_candidates] OF REAL	1 dimensional array used to store the percentage of votes for each candidate in election
highest_votes	integer	DECLARE highest_votes: INTEGER	variable used to find the candidate with the highest number of votes
candidate_winner	string	DECLARE candidate_winner: STRING	variable used to find the name of the winner of the election
no_of_winners	integer	DECLARE no_of_winners: INTEGER	variable used to find the number of 'winners' in the election, especially when it's a tie
tied_candidates	1D array	DECLARE tied_candidates: ARRAY [1 TO names[count]] OF STRING	1 dimensional array used to find the names of the tied candidates if this occurs in election

PSEUDOCODE:

```
START

names ← []
tied_candidates ← []
student_abstention ← 0
highest_votes ← 0
total_votes ← 0
no_of_winners ← 0

OUTPUT "please input the name of your tutor group"
INPUT tutor_group

OUTPUT "please enter number of students in your tutor group"
INPUT num_students

WHILE num_students < 28 OR num_students > 35 DO
    OUTPUT "error out of range try again"
    OUTPUT "please enter number of students in your tutor group"
    INPUT num_students
ENDWHILE

OUTPUT "please enter number of candidates in election"
INPUT num_candidates

WHILE num_candidates < 1 OR num_candidates > 4 DO
    OUTPUT "error out of range try again"
    OUTPUT "please enter number of candidates in election"
    INPUT num_candidates
ENDWHILE

vote_count ← [FOR count = 1 TO num_candidates]

FOR count ← num_candidates
    OUTPUT "please input names of candidates in election"
    INPUT candidate_names
    names.append(candidate_names)
NEXT count

student_unique ← [FOR count = 1 TO num_students]

WHILE TRUE DO:

    FOR count ← num_students
        OUTPUT "input unique number"
        INPUT unique_num
        OUTPUT unique_num
```

```
     IF unique_num IN student_unique
         THEN
           OUTPUT "voted and cannot vote again"
         ELSE
           student_unique[count] = unique_num
           OUTPUT "vote or abstain"
           INPUT student_choice

           IF student_choice == "abstain"
               THEN
                 student_abstention ← student_abstention + 1

          IF student_choice == "vote"
               THEN
                 OUTPUT "who would you like to vote for?"
                 INPUT candidate_votes

                 IF candidate_votes == 1
                     THEN
                       vote_count[1] = vote_count [1] + 1

                 IF candidate_votes == 2
                     THEN
                       vote_count[2] = vote_count [2] + 1

                 IF candidate_votes == 3
                     THEN
                       vote_count[3] = vote_count [3] + 1

                 IF candidate_votes == 4
                     THEN
                       vote_count[4] = vote_count [4] + 1
                 ENDIF
          ENDIF
    ENDIF
NEXT count

FOR count ← LEN (vote_count)
    total_votes ←  num_students - student_abstention
NEXT count

percentage_votes ← [FOR count = 1 TO num_candidates]

FOR count ← num_candidates
    percentage_votes[count] ← (vote_count[count] / total_votes)*100
NEXT count

OUTPUT "Tutor group", tutor_group
OUTPUT "names of candidates and their vote counts and percentages are:"
OUTPUT names
OUTPUT vote_count
OUTPUT percentage_votes "%"
```

```
    OUTPUT "total votes:", total_votes
    OUTPUT "total abstentions:", student_abstention

    FOR count ← num_candidates
        IF vote_count[count] > highest_votes
            THEN
              highest_votes ← vote_count
        ENDIF
    NEXT count



    FOR count ← num_candidates
        IF vote_count[count] == highest_votes
            THEN
              candidate_winner ← names[count]
              no_of_winners ← no_of_winners + 1
        ENDIF
    NEXT count

    IF no_of_winners > 1
        THEN
          FOR count ← num_candidates
              IF vote_count[count] == highest_votes
                  THEN
                    tied_candidates.append(names[count])
                    num_candidates ← no_of_winners
               ENDIF

          OUTPUT "we have a tie between:", tied_candidates
          names ←  tied_candidates
          student_unique ←  [FOR count = 1 TO num_students]
          vote_count ←  [FOR count = 1 TO num_candidates]
          student_abstention ← 0
          highest_votes ←  0
          no_of_winners ←  0
          NEXT count

        ELSE
          OUTPUT "the winner is:", candidate_winner
          BREAK
    ENDIF
ENDWHILE
END
```

TEST PLAN:

num_candidates

Test type	Test value	Expected outcome	Actual outcome	Description
Normal	2,3	**2, 3**		Data which is acceptable within the given range.
Abnormal	“f”, 55	**error message**		Data which is unacceptable within the given range and rejected and displays error message.
Extreme	1, 4	**1, 4**		Data and the end of the given range- acceptable.

num_students

Test type	Test value	Expected outcome	Actual outcome	Description
Normal	29,32	**29, 32**		Data which is acceptable within the given range.
Abnormal	“two”, 73	**error message**		Data which is unacceptable within the given range and rejected and displays error message.
Extreme	28, 35	**28, 35**		Data and the end of the given range- acceptable.

ANNOTATED PROGRAM LISTING:

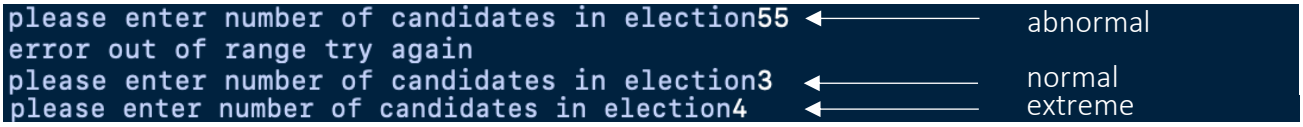
```
#pre release 2021
#pre release voting system


names = []
tied_candidates = []
student_abstention = 0
highest_votes = 0
total_votes = 0
no_of_winners = 0

#allowing the user to input name of tutor group
tutor_group = input("please input the name of your tutor group")

#allowing the user to input number of students in tutor group
num_students = int(input("please enter number of students in your tutor group"))
while num_students <28 or num_students >35:
    print ("error out of range try again")
    num_students= int(input("please enter number of students your tutor group"))

#allowing the user to input number of candidates in election
num_candidates = int(input("please enter number of candidates in election"))

#making sure the number of candidates inputted is between the acceptable range
while num_candidates <1 or num_candidates >4:
    print ("error out of range try again")
    num_candidates = int(input("please enter number of candidates in election"))

vote_count = [0 for count in range(num_candidates)]

#allowing the user to input names of candidates in election
for count in range (num_candidates):
    candidate_names = str(input("input name of candidate in election"))
    names.append(candidate_names)
student_unique = [0 for count in range(num_students)]

while True:

#allowing the user to input their unique number
    for count in range (num_students):
        unique_num = int(input("input unique number"))
        print(unique_num)

# making sure the user doesnt input the same number to vote twice
        if unique_num in student_unique:
            print ("voted and cannot vote again")

        else:
            student_unique[count] = unique_num
            student_choice = input("vote or abstain")

#incrementing the number of abstentions
            if student_choice == "abstain":
                student_abstention = student_abstention + 1

#asking the user which candidate they would like to vote for
            if student_choice == "vote":
                candidate_votes = int(input("who would you like to vote for?"))

                if candidate_votes == 1:
                    vote_count[0] = vote_count[0]+ 1

                if candidate_votes == 2:
                    vote_count[1] = vote_count[1]+ 1

                if candidate_votes == 3:
                    vote_count[2] = vote_count[2]+ 1

                if candidate_votes == 4:
                    vote_count[3] = vote_count[3]+ 1
```

```
#calculating the total number of votes in the election
    for count in range (len(vote_count)):
        total_votes = num_students - student_abstention

    percentage_votes = [0 for count in range(num_candidates)]

#calculating the percentage votes of each candidate
    for count in range (num_candidates):
        percentage_votes[count] = (vote_count[count] / total_votes)*100

#printing all the relevant information from above
    print ("Tutor group",tutor_group)
    print ("names of candidates and their vote counts and percentages are:")
    print (names)
    print (vote_count)
    print (percentage_votes, "%")
    print ("total votes:", total_votes)
    print ("total abstentions:", student_abstention)

#finding which candidate has got the highest number of votes
    for count in range (num_candidates):
        if vote_count[count] > highest_votes:
            highest_votes = vote_count[count]

#finding the candidate winner of the election or finding a tie
    for count in range (num_candidates):
        if vote_count[count] == highest_votes:
            candidate_winner = names[count]
            no_of_winners += 1

#printing the names of candidates with a tie and resetting the variables
    if no_of_winners > 1:
        for count in range (num_candidates):
            if vote_count[count] == highest_votes:
                tied_candidates.append(names[count])
                num_candidates = no_of_winners
        print ("we have a tie between:",tied_candidates)

        names = tied_candidates
        student_unique = [0 for count in range(num_students)]
        vote_count = [0 for count in range(num_candidates)]
        student_abstention = 0
        highest_votes = 0
        no_of_winners = 0

#printing the name of the winner of the election
    else:
        print ("the winner is:", candidate_winner)
        break
```

TEST RESULTS:

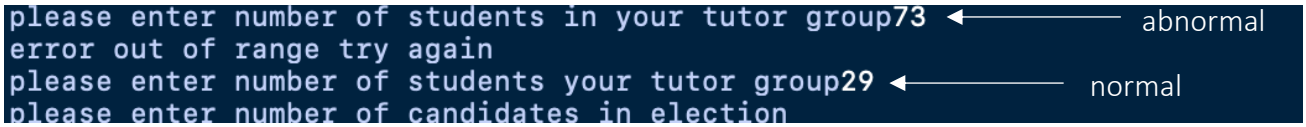
num_candidates

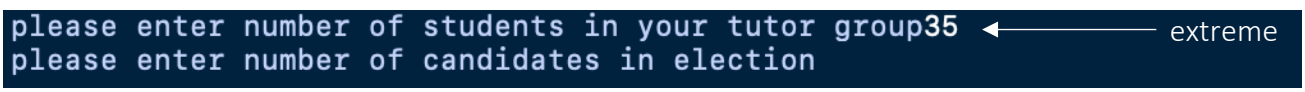
Test type	Test value	Expected outcome	Actual outcome	Description
Normal	2,3	2, 3	2,3	Data which is acceptable within the given range.
Abnormal	"f", 55	error message	error message	Data which is unacceptable within the given range and rejected and displays error message.
Extreme	1, 4	1, 4	1,4	Data and the end of the given range- acceptable.



num_students

Test type	Test value	Expected outcome	Actual outcome	Description
Normal	29,32	29, 32	29, 32	Data which is acceptable within the given range.
Abnormal	"two", 73	error message	error message	Data which is unacceptable within the given range and rejected and displays error message.
Extreme	28, 35	28, 35	28, 35	Data and the end of the given range- acceptable.





QUESTIONS IN RELATION TO PROGRAM:

a) All variables , constants and other identifiers should have meaningful names, using CIE standards show how you complete the following:

Declare the array to store student unique number [1]
student_unique

Declare the arrays to store the candidate names [1]
names (for all candidate names)
tied_candidates (for tied candidate names)

b) Discuss with code examples how you validated the student numbers in each class [3]

Firstly, I used a while loop to make sure the range of the number of students inputted was within 28 to 35. Next, if the integer inputted was out of this range, the program displayed an error message and the user is asked again to input the number.

```
#allowing the user to input number of students in tutor group
num_students = int(input("please enter number of students in your tutor group"))
while num_students <28 or num_students>35:
    print ("error out of range try again")
    num_candidates = int(input("please enter number of students your tutor group"))
```

c) Explain how you store the votes for the students and how you compared if a student already voted? You must include programming statements or pseudo code to help you explain. [5]

Firstly, I created an array to store all the unique numbers the user inputs depending of the number of students. Then, I asked the user to input each students unique number to vote with and it was stored in this array. If the user inputted the same number as one of previous numbers stored in the array, an error message is displayed and the user is asked again.

```
student_unique = [0 for count in range(num_students)]
```

```
#allowing the user to input their unique number
    for count in range (num_students):
        unique_num = int(input("input unique number"))
        print(unique_num)

# making sure the user doesnt input the same number to vote twice
        if unique_num in student_unique:
            print ("voted and cannot vote again")
```

d) Explain how you identify the percentage of votes obtained for each candidate. You must include programming statements or pseudo code to help you explain. [5]

Firstly, I calculated the total votes. Then I created an array to store all the percentages of votes for each candidate in the range of the number of candidates in the election. Next, I calculated the percentages by taking the vote count of each candidate and dividing it over the total votes, then multiplying by 100.

```
#calculating the total number of votes in the election
    for count in range (len(vote_count)):
        total_votes = num_students - student_abstention

    percentage_votes = [0 for count in range(num_candidates)]

#calculating the percentage votes of each candidate
    for count in range (num_candidates):
        percentage_votes[count] = (vote_count[count] / total_votes)*100
```

e) If a tie explain how your code allows re-voting for just the Tied candidates. You must include programming statements or pseudo code to help you explain. [5]

To revote for tied candidates, I created a while loop at the start to repeat the program if this occurs, but only with the names of the tied candidates. I created an array for the names of tied candidates and if the number of "winners" was greater than 1 and the number of votes equaled the highest votes, I appended the names of the candidates with a tie. Then the previous names list only included the tied candidates and I assigned 0 to all the necessary variables to make sure nothing from the last voting system was repeated. Program breaks if there is only 1 winner.

```
while True:
```

program continues

```
#printing the names of candidates with a tie and resetting the variables
    if no_of_winners > 1:
        for count in range (num_candidates):
            if vote_count[count] == highest_votes:
                tied_candidates.append(names[count])
                num_candidates = no_of_winners
        print ("we have a tie between:",tied_candidates)

        names = tied_candidates
        student_unique = [0 for count in range(num_students)]
        vote_count = [0 for count in range(num_candidates)]
        student_abstention = 0
        highest_votes = 0
        no_of_winners = 0
```

f) Show two different sets of test data that you could use to check the validation you used in task 1 of your program [4]

For validating number of candidates:
Set 1: normal e.g. 2, 3
Reason for choice Data which is acceptable within the given range.

Set 2 abnormal e.g. "three", 42
Reason for choice Data that is unaccepted within given range, rejected and displays error message