Assignment #4

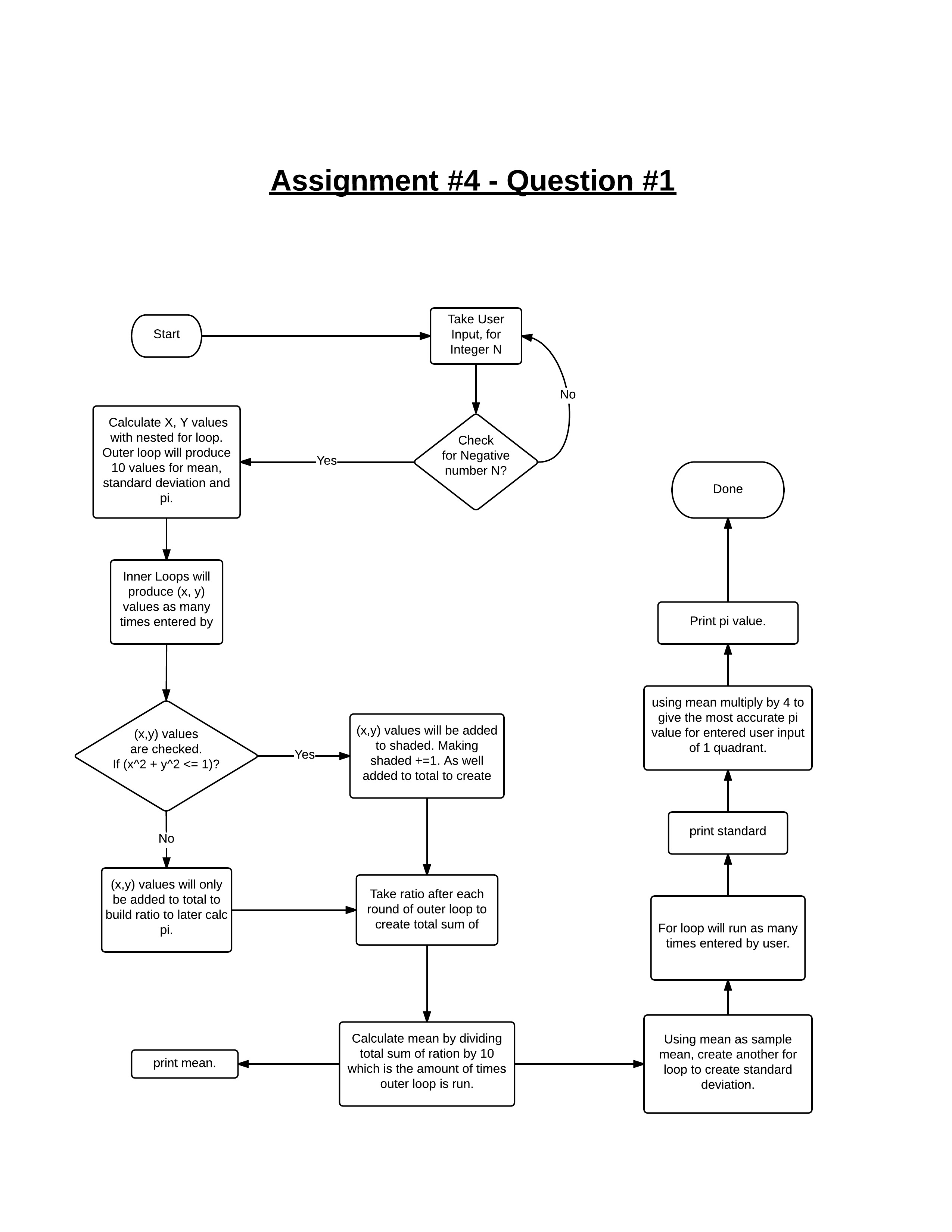
By: Josh Jackson

Question #1

Code:

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <time.h>  #include <math.h>  #define TRUE 0  #define FALSE 1  int main(){  //declare variables  int n;  int flag;  //total x,y values and amount of x,y values in shaded region  float total = 0.0;  float shade = 0.0;    //do-while statment to check if user input is a positive integer  do {  printf("Enter N: \n");  scanf("%d", &n);  if (n < 0) {  printf("Invalid Entry, Try Again..\n");  flag = 1;  }else flag = 0;    } while (flag == 1);    /\*  using time to create constant to define srand() function which will  initizialize rand() function and create different numbers each time  \*/  time\_t t;  srand((unsigned) (time(&t)));    float x = 0.0, y = 0.0;  float pi = 0.0;  float ratio = 0.0;  float mean = 0.0;  float std\_dev = 0.0;  //X and Y value loop which will calculate ratio and mean 10 times  for (int i = 0; i < 10; i++) {  //inner loops will calculate the x and y values based on input from user  for (int i = 0; i < n; i++) {  float k = (rand() % 100);  k = k/100;  x = k;    for (int i = 0; i < n; i++) {  float k = (rand() % 100);  k = k/100;  y = k;  }  //if statement to count the valid values in shaded region and total values  if ((x\*x)+(y\*y) <= 1) {  shade++;  total++;  }else  total ++;  }  ratio = (shade / total);  printf("Values: %f\n", ratio);  mean += ratio;    }  //Standard Deviation For-Loop  for (int i = 1; i <= n; i++) {  int s = n;  std\_dev = ((i - mean)/s);  }  //standard deviation square rooot  std\_dev = sqrtf(std\_dev);  //mean calculation  mean /= 10;  //ration calculation  ratio = (shade / total);  //most accurate pi calculation within in 1 quadrant  pi = mean\*4.0;    //printing values for mean, standard dev, and pi  printf("Mean = %f\n", mean);  printf("Standard Deviation = %f\n", std\_dev);  printf("Pi = %f\n", pi);    //printf("Ratio = %f\n", ratio);  } |

Flow Chart:



Test Cases:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | Test |  |  | | Input | Output |
| 1 | **N = 10** | **Values: 0.90**  **Values: 0.80**  **Values: 0.77**  **Values: 0.75**  **Values: 0.76**  **Values: 0.80**  **Values: 0.83**  **Values: 0.80**  **Values: 0.82**  **Values: 0.82**  **Mean = 0.80**  **Standard Deviation = 0.44**  **Pi = 3.218984** |
| 2 | **N = 100** | **Enter N: 100**  **Values: 0.83**  **Values: 0.83**  **Values: 0.82**  **Values: 0.82**  **Values: 0.83**  **Values: 0.82**  **Values: 0.82**  **Values: 0.81**  **Values: 0.80**  **Values: 0.80**  **Mean = 0.82**  **Standard Deviation = 0.96**  **Pi = 3.271564** |
| 3 | **N = 1000** | **Enter N: 1000**  **Values: 0.79**  **Values: 0.80**  **Values: 0.79**  **Values: 0.80**  **Values: 0.79**  **Values: 0.79**  **Values: 0.80**  **Values: 0.80**  **Values: 0.80**  **Values: 0.80**  **Mean = 0.79**  **Standard Deviation = 1.00**  **Pi = 3.178802** |
| 4 | **N = 10 000** | **Enter N: 10000**  **Values: 0.79**  **Values: 0.79**  **Values: 0.79**  **Values: 0.79**  **Values: 0.80**  **Values: 0.79**  **Values: 0.80**  **Values: 0.80**  **Values: 0.80**  **Values: 0.80**  **Mean = 0.79**  **Standard Deviation = 1.00**  **Pi = 3.178880** |
| 5 | **N = 100 000** | **Enter N: 100000**  **Values: 0.79**  **Values: 0.79**  **Values: 0.79**  **Values: 0.79**  **Values: 0.79**  **Values: 0.79**  **Values: 0.79**  **Values: 0.79**  **Values: 0.79**  **Values: 0.79**  **Mean = 0.79**  **Standard Deviation = 0.99**  **Pi = 3.179937** |
| 6 | **N = 1 000 000** | **Enter N: 1000000**  **Values: 0.79**  **Values: 0.79**  **Values: 0.79**  **Values: 0.79**  **Values: 0.79**  **Values: 0.79**  **Values: 0.79**  **Values: 0.79**  **Values: 0.79**  **Values: 0.79**  **Mean = 0.79**  **Standard Deviation = 1.00**  **Pi = 3.166562** |
| 7 | **N = 10 000 000** | **Enter N:** 10000000  **Values: 0.80**  **Values: 0.80**  **Values: 0.80**  **Values: 0.80**  **Values: 0.80**  **Values: 0.80**  **Values: 0.80**  **Values: 0.80**  **Values: 0.80**  **Values: 0.80**  **Mean = 0.80**  **Standard Deviation = 1.00**  **Pi = 3.187313** |
| 8 | **N = 100 000 000** | **Enter N: 10000000**  **Values: 0.79**  **Values: 0.80**  **Values: 0.80**  **Values: 0.80**  **Values: 0.80**  **Values: 0.80**  **Values: 0.80**  **Values: 0.79**  **Values: 0.79**  **Values: 0.80**  **Mean = 0.79**  **Standard Deviation = 1.00**  **Pi = 3.179758** |

Discuss

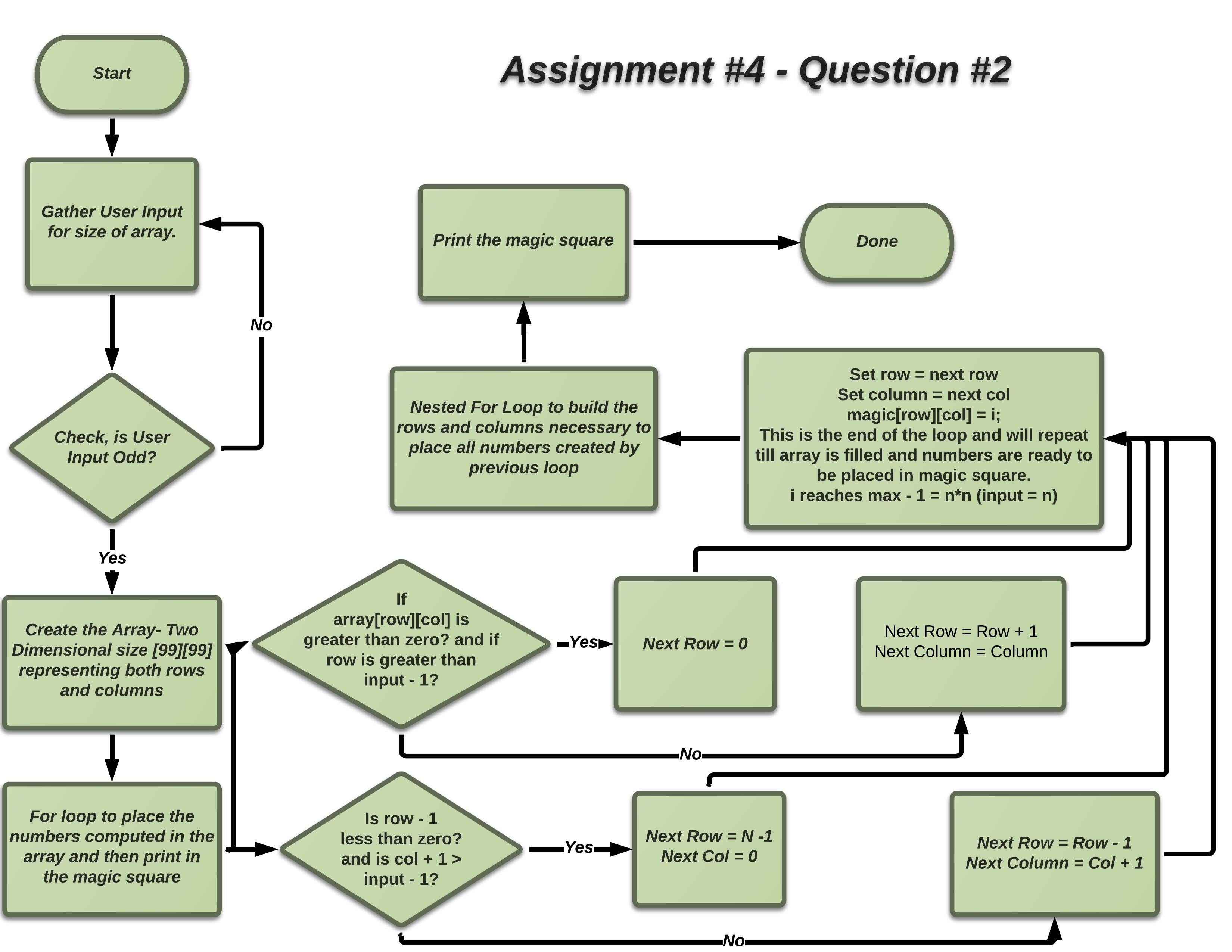
* It seems the higher the value of N the more consistent the ratio becomes 0.79 making the mean become 0.79. As well the more values of (x, y) the more accurate Pi value becomes. Standard deviation also becomes much closer to 1.00 the more values it has to compute. As we can see for N = 10 where standard deviation is very different compared to N = 100 000.

Question 2

Code:

|  |
| --- |
| #include <stdio.h>  #define FALSE 1  #define TRUE 0  int main() {  int flag;//for true or false to be checked when number is entered  int n;  flag = 1;//setting to false to enter do-while  do {  printf("Enter size of magic square: \n");  scanf("%d", &n);  //if number is even it will repeat loop else exits  if (n % 2 == 0) {  printf("Invalid size, try again...\n");  flag = 1;  }else flag = 0;    } while (flag == 1);  // Get the users magic number and allocate to int n    int magic[99][99];// Create the array size of 99 rows and 99 columns  int start = (n / 2); // The middle column    int max = n \* n; // The final number to be computed    magic[0][start] = 1; // Place the number one in the middle of row 0    // Loop to start placing numbers in the magic square  int row;  int column;  int next\_row;  int next\_column;  int i;  for (i = 2, row = 0, column = start; i < max + 1; i++) {    if ((row - 1) < 0) { // If going up one will leave the top level of square  next\_row = n - 1; // enter number in bottom row  }  else { next\_row = row - 1; } //if not go up one row    if ((column + 1) > (n - 1)) { // If column will leave the farthest side of square  next\_column = 0; // Wrap back to first column  }  else { next\_column = column + 1; } // Otherwise go over one column    if (magic[next\_row][next\_column] > 0) { // If next number to be entered's position is full  if (row > (n - 1)) { // If going to row below leaves bottom  next\_row = 0; // Go back to the top  }  else {  next\_row = row + 1; // allocates number to next row  next\_column = column; // But stay in same column  }  }  row = next\_row;  column = next\_column;  magic[row][column] = i; // Put the current value in that position  //continue this process tille i reaches max-1 = n\*n  //then exit into next loop where it builds and prints the size of the array with computed values  }    // Now let's print the array  int j;  for (i = 0; i < n; i++) {  for (j = 0; j < n; j++) {  printf("%4d", magic[i][j]);  }  printf("\n");  }  return 0;  } |

Flow Chart:



Test Cases

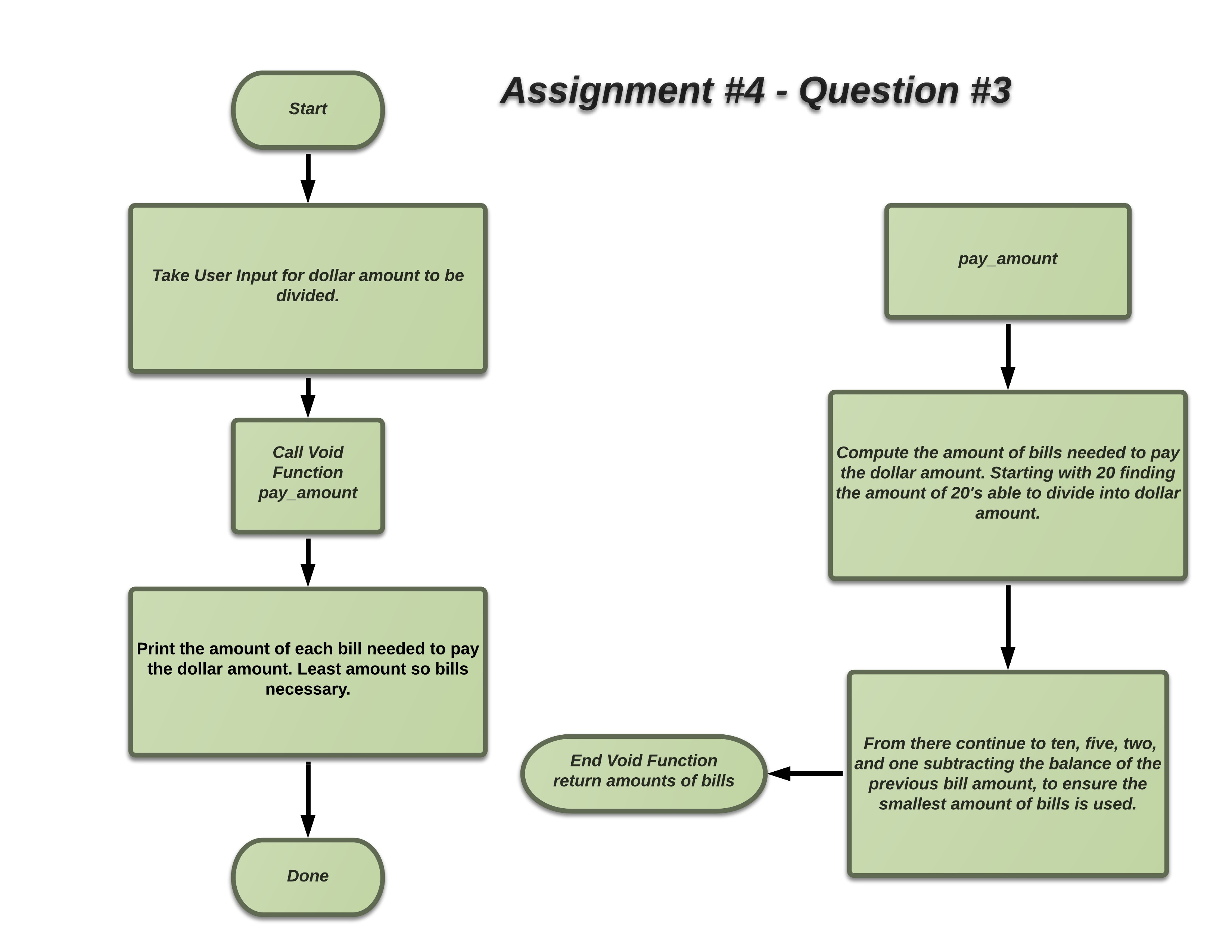
|  |  |  |
| --- | --- | --- |
| Test | Input | Output |
| 1 | **Enter size of magic square:**  3 | **8 1 6**  **3 5 7**  **4 9 2** |
| 2 | **Enter size of magic square:**  5 | **17 24 1 8 15**  **23 5 7 14 16**  **4 6 13 20 22**  **10 12 19 21 3**  **11 18 25 2 9** |
| 3 | **Enter size of magic square:**  4 | **Invalid size, try again...**  **Enter size of magic square:** |
| 4 | **Enter size of magic square:**  9 | **47 58 69 80 1 12 23 34 45**  **57 68 79 9 11 22 33 44 46**  **67 78 8 10 21 32 43 54 56**  **77 7 18 20 31 42 53 55 66**  **6 17 19 30 41 52 63 65 76**  **16 27 29 40 51 62 64 75 5**  **26 28 39 50 61 72 74 4 15**  **36 38 49 60 71 73 3 14 25**  **37 48 59 70 81 2 13 24 35** |

Question 3

Code:

|  |
| --- |
| #include <stdio.h>  #define TRUE 0  #define FALSE 1  int dollars;  int twenties;  int tens;  int fives;  int toonies;  int loonies;  int total;  //void function declaration  void pay\_amount(int dollars, int \*twenties, int \*tens, int \*fives, int \*toonies, int \*loonie);  //pay\_amount function  void pay\_amount(){    //computing each variable according amount of bill  twenties = dollars / 20;  tens = (dollars - (twenties \* 20)) /10;  fives = (dollars - (twenties\*20) - (tens\*10)) /5;  toonies = (dollars - (twenties\*20) - (tens\*10) - (fives\*5)) /2;  loonies = (dollars - (twenties\*20) - (tens\*10) - (fives\*5) - (toonies\*2)) / 1;  total = twenties + tens + fives + toonies + loonies;  }  //main function  int main(void){  int flag;  //printing and scanning dollar amount to be computed from user  do {  printf("Enter a Dollar Amount: \n");  scanf("%d", &dollars);  if (dollars < 0) {  printf("Invalid Dollar Amount.. \n");  flag = 1;    }else flag = 0;  } while (flag == 1);    //calling void function to gain access to computations  pay\_amount(dollars, &twenties, &tens, &fives, &toonies, &loonies);    //printing out all values  //total amount enters by user  printf("Total Dollar Amount is equal to $%d\n", dollars);  //smallest amount of bills with ability to pay amount  printf("Smallest amount of bills to pay is %d bill(s).\n", total);  //total twenty dollar bills needed  printf("%d: Twenty Dollar Bill(s)\n", twenties);  //total ten dollar bills  printf("%d: Ten Dollar Bill(s)\n", tens);  //total five dollar bills  printf("%d: Five Dollar Bill(s)\n", fives);  //total two dollar bills  printf("%d: Two Dollar Bill(s)\n", toonies);  //total one dollar bills  printf("%d: One Dollar Bill(s) \n", loonies);  } |

Flow Chart:



Test Cases:

|  |  |  |
| --- | --- | --- |
| Test | Input | Output |
| 1 | **Enter a Dollar Amount:**  77 | **Total Dollar Amount is equal to $77**  **Smallest amount of bills to pay is 6 bill(s).**  **3: Twenty Dollar Bill(s)**  **1: Ten Dollar Bill(s)**  **1: Five Dollar Bill(s)**  **1: Two Dollar Bill(s)**  **0: One Dollar Bill(s)** |
| 2 | **Enter a Dollar Amount:**  1 | **Total Dollar Amount is equal to $1**  **Smallest amount of bills to pay is 1 bill(s).**  **0: Twenty Dollar Bill(s)**  **0: Ten Dollar Bill(s)**  **0: Five Dollar Bill(s)**  **0: Two Dollar Bill(s)**  **1: One Dollar Bill(s)** |
| 3 | **Enter a Dollar Amount:**  -123 | **Invalid Dollar Amount...**  **Enter a Dollar Amount:** |
| 4 | **Enter a Dollar Amount:**  .60 | **Total Dollar Amount is equal to $0**  **Smallest amount of bills to pay is 0 bill(s).**  **0: Twenty Dollar Bill(s)**  **0: Ten Dollar Bill(s)**  **0: Five Dollar Bill(s)**  **0: Two Dollar Bill(s)**  **0: One Dollar Bill(s)** |
| 5 | **Enter a Dollar Amount:**  60 | **Total Dollar Amount is equal to $60**  **Smallest amount of bills to pay is 3 bill(s).**  **3: Twenty Dollar Bill(s)**  **0: Ten Dollar Bill(s)**  **0: Five Dollar Bill(s)**  **0: Two Dollar Bill(s)**  **0: One Dollar Bill(s)** |