Question 1 ---

PROBLEM - Automatic Teller Machine that dispense money.

ANALYSIS - To Solve this problem, first we need to get the desired amount which should be multiple of 10.

Finally, It should then display the bill dispense by calculating the number of 50s , 20s and 10s.

DATA REQUIREMENTS:

Problem Inputs -

int dollar;

Problem Outputs -

int fifty , twenty , ten;

DESIGN:

INITIAL ALGORITHM -

Step1. Read the desired amount (multiple of 10)

Step2. Store the amount

Step3. check if the amount is multiple of 10 otherwise print number is not a multiple of 10

Step4. Using the bill_dispense function calculating the number of 50s, 20s, 10s.

Step5. Displaying the bill dispense.

IMPLEMENTATION -

```
#include <stdio.h>
int bill_dispense(int , int* , int* , int* );
int main() {
   int dollar, fifty = 0, twenty = 0, ten = 0;
   printf("Enter the amount desired (a multiple of 10 dollars) - ");
   scanf("%d",&dollar);
   if((dollar % 10) == 0){
      printf("Bill to pay : \n");
```

```
        bill_dispense(dollar , &fifty , &twenty , &ten);
        printf("50s are %d\n",fifty);
        printf("20s are %d\n",twenty);
        printf("10s are %d\n",ten);
    }
    else{
        printf("Entered number is not a multiple of 10");
    }

    return 0;
}
int bill_dispense(int dollar , int *fifty , int *twenty , int *ten){
    *fifty = dollar/50;
    dollar = dollar %50;
    *twenty = dollar/20;
    dollar  = dollar%20;
    *ten = dollar/10;
    dollar = dollar%10;
}
```

Output -

```
Enter the amount desired (a multiple of 10 dollars) - 475
Entered number is not a multiple of 10

Enter the amount desired (a multiple of 10 dollars) - 280
Bill to pay :
50s are 5
20s are 1
10s are 1
```

Question 2 ---

PROBLEM - Program for dispense change.

ANALYSIS - To Solve this problem, first we need to get the amount paid and amount due by the user.

Finally, It should then display the by executing the function cal_change() the change given in dollars, pennies , dimes, nickels , quarters.

DATA REQUIREMENTS:

Problem Inputs -

float amt_paid, amt_due;

Problem Outputs -

int dollars, pennies, dimes, quarters , nickels.

DESIGN:

INITIAL ALGORITHM -

```
Step1. Read the amount paid.
Step2. Store the amout paid.
Step3. Read the amount dues.
Step4. Store the amount dues.
Step5. Calculating the change by calling the function cal_change()
Step6. Displaying the how much change is given in dollars , pennies, dimes, quarters, nickels.
```

IMPLEMENTATION -

```
#include <stdio.h>
int main() {
   float amt_paid, amt_due;
   int dollars, quarters, dimes, nickels, pennies;
   // Get the input from the user
   printf("Enter the amount paid: $");
```

```
    scanf("%f", &amt_paid);
    printf("Enter the amount due: $");
    scanf("%f", &amt_due);
    // Calculate the change amount in cents to avoid floating-point precision issues
     int changeInCents = (int)((amt_paid - amt_due) * 100);
    // Call the function to calculate change
    cal_change(changeInCents, &dollars, &quarters, &dimes, &nickels, &pennies);
    // Display the results
    printf("Change to be dispensed:\n");
    printf("Dollars: %d\n", dollars);
    printf("Quarters: %d\n", quarters);
    printf("Dimes: %d\n", dimes);
    printf("Nickels: %d\n", nickels);
    printf("Pennies: %d\n", pennies);
    return 0;
}
void cal_change(int changeInCents, int* dollars, int* quarters, int* dimes, int* nickels,
 int* pennies) {
    // Calculate the number of dollars
    *dollars = changeInCents / 100;
    changeInCents %= 100;
    // Calculate the number of quarters
    *quarters = changeInCents / 25;
    changeInCents %= 25;
    // Calculate the number of dimes
    *dimes = changeInCents / 10;
    changeInCents %= 10;
    // Calculate the number of nickels
    *nickels = changeInCents / 5;
    changeInCents %= 5;
    // The remaining changeInCents is in pennies
    *pennies = changeInCents;
}
```

Output -

```
Enter the amount paid: $200
Enter the amount due: $128.34
Change to be dispensed:
Dollars: 71
Quarters: 2
Dimes: 1
Nickels: 1
Pennies: 1
```

Question 3 ---

PROBLEM - Checking for multiple of 7,11,13 and is the sum of digit is odd or even and
 check of prime number.

ANALYSIS - To Solve this problem, first we need to get the number from the user.
Finally, It should then display the weather the number is multiple of 7,11,13 or sum of
digit is either even or odd or
the given number is prime number.

DATA REQUIREMENTS:

Problem Inputs -

int num;

Problem Outputs -

int q1,q2,q3;

DESIGN:

INITIAL ALGORITHM -

```
Step1. Read the number.
Step2. Store the number.
Step3. Check for the condition :
a. Is the value a multiple of 7, 11, or 13?
b. Is the sum of the digits odd or even?
c. Is the value a prime number?
Step4. Printing the information.
```

IMPLEMENTATION -

```c
#include <stdio.h>
#include <math.h>
int check(int, int*, int*, int*);
int digits_odd(int);
int isprime(int);
int main()
  {
  int num, q1=0, q2=0, q3=0;
  printf("Enter a number-> ");
  scanf("%d", &num);
  check(num, &q1, &q2, &q3);
  if(q1 == 1)
    printf("\nThe number is a multiple of 7, 11, or 13.");
  else
    printf("\nThe number is not a multiple of 7, 11, or 13.");
  if(q2 == 1)
   printf("\nThe sum of the digits is odd.");
  else
    printf("\nThe sum of the digits is even.");
  if(q3 == 1)
```

```
        printf("\nThe number is a prime number.\n");
    else
        printf("\nThe number is not a prime number.\n");
    return 0;
}
int check(int num, int *q1, int *q2, int *q3)
{
    if(((num % 7) == 0) || ((num % 11) == 0) || ((num % 13) == 0))
       *q1 = 1;
    if(digits_odd(num))
        *q2 = 1;
    if(isprime(num))
        *q3 = 1;
}
int digits_odd(int num)
{
    int temp;
    int total = 0;
    if(num == 0)
        total = 0;
    else
    {
        while (num > 0)
        {
            temp = num % 10;
            total += temp;
            num = num / 10;
        }
    }
    if(total % 2 == 0)
        return 0;
    return 1;
}
int isprime(int n)
{
    int i, flag = 1;
    for(i = 2; i <= sqrt(n); i++)
    {
        if(n % i == 0)
        {
            flag = 0;
            break;
        }
    }
return flag;
}
```

Output -

```
Enter a number-> 47
The number is not a multiple of 7, 11, or 13.
The sum of the digits is odd.
The number is a prime number.
```

Question 4 ---

PROBLEM - Calulating SquareRoot of a number.

ANALYSIS - To Solve this problem, first we need to get the number and assuming the initial guess i.e LG = 1.0.

Finally, It should then display the SquareRoot of the number.

DATA REQUIREMENTS:

Problem Inputs -

double n;

Problem Outputs -

doube NG;

DESIGN:

INITIAL ALGORITHM -

Step1. Read the number.

Step2. Store the number.

Step3. Calculating the square root of a number using approximate_square_root().

Step4. Printing the square root.

IMPLEMENTATION -

```
#include <stdio.h>

#include <math.h>

#define DIFFERENCE 0.005

void approximate_square_root(double, double, double*);

void main()

{

   double n, LG = 1.0, NG;

   n = 4;

   printf("\nNumber - %f", n);

   approximate_square_root(n, LG, &NG);

   printf("\nSquare root - %f", NG);

   n = 120.5;

   printf("\n\nNumber - %f", n);

   approximate_square_root(n, LG, &NG);

   printf("\nSquare root - %f", NG);

   n = 88;

   printf("\n\nNumber - %f", n);

   approximate_square_root(n, LG, &NG);

   printf("\nSquare root - %f", NG);
```

```
    n = 36.01;
    printf("\n\nNumber - %f", n);
    approximate_square_root(n, LG, &NG);
    printf("\nSquare root - %f", NG);
    n = 10000;
    printf("\n\nNumber - %f", n);
    approximate_square_root(n, LG, &NG);
    printf("\nSquare root - %f", NG);
    n = 0.25;
    printf("\n\nNumber - %f", n);
    approximate_square_root(n, LG, &NG);
    printf("\nSquare root - %f", NG);
    printf("\n");
}
void approximate_square_root(double N, double LG, double *NG)
{
    *NG = 0.5 * (LG + (N / LG));
    while(fabs(*NG - LG) > DIFFERENCE)
    {
        LG = *NG;
        *NG = 0.5 * (LG + (N / LG));
    }
}
```

Output -

```
Number - 4.000000
Square root - 2.000000

Number - 120.500000
Square root - 10.977249
```

```
Number - 88.000000
Square root - 9.380832

Number - 36.010000
Square root - 6.000833

Number - 10000.000000
Square root - 100.000000

Number - 0.250000
Square root - 0.500000
```

Question 5 ---

PROBLEM - Calculation of Drag Force of an Aircraft or an Automobile
 which is moving through the atmosphere.

ANALYSIS - To Solve this problem, first we need to get the projected
 area and the drag coefficient.

Finally, It should then display the drag_force for the range of velocities.

DATA REQUIREMENTS:

Problem Inputs -

double A, CD;

Problem Outputs -

double F;

DESIGN:

INITIAL ALGORITHM -

```
Step1. Read the area.
Step2. Store the area.
```

Step3. Read the drag-coefficient.

Step4. Store the drag-coefficient.

Step5. Running the loop for velocity from 0 to 40 m/s with increament of 5m/s.

Step6. Calculating and the drag_force.

IMPLEMENTATION -

```
#include <stdio.h>
#include <math.h>
#define RHO 1.23
double calculate(double, double, int, double*);
int main(){
   double A , CD, F;
   printf("Enter the Area(A) - ");
   scanf("%lf",&A);
   printf("Enter the drag-coefficient(CD) - ");
   scanf("%lf",&CD);
   printf("\n Velocity %5c Drag_Force\n");
   for(int V = 0 ; V < 40 ; V +=5){
      calculate(A, CD, V, &F);
      printf("%dm/s %5c %0.2fN\n",V,F);
   }
   return 0;
}
double calculate(double A , double CD, int V, double *F){
   *F = 0.5 * CD * A * RHO * pow(V, 2);
}
```

Output -

```
Enter the Area(A) - 100
Enter the draf-coefficient(CD) - 0.3
Velocity     Drag_Force
```

0m/s	0.00N
5m/s	461.25N
10m/s	1845.00N
15m/s	4151.25N
20m/s	7380.00N
25m/s	11531.25N
30m/s	16605.00N
35m/s	22601.25N

Question 6 ---

PROBLEM - Finding the final approimation of e and value of e.

ANALYSIS - To Solve this problem, first we need to get the value of x,checking if the
absolute difference
is less than or greater than 0.000001, if less then loop terminates.
Finally, It would print approx value of e and the value of e calulated using the expression.

DATA REQUIREMENTS:

Problem Inputs -

int x;

Problem Outputs -

double approx;

DESIGN:

INITIAL ALGORITHM -

Step1. Reading the value of x = 1 at starting.

Step2. Assuming the value of diff = 1.

Step3. checking if the absolute difference is greater than or equal to 0.000001
otherwise loop terminate.

Step4. calulating the approx value of e using the approximate() function.

Step5. increamenting the value of x until while loop gets terminated.

Step6. printing the approx value of e using the expression.

Step7. printing the exponentional value of e.

IMPLEMENTATION -

```
#include <stdio.h>
#include <math.h>
double approximate(int, double*);
int main()
{
   int x = 1;
   double approx, diff = 1;
   while(diff >= 0.000001)
   {
      approximate(x, &approx);
      diff = fabs(exp(1) - approx);
      x++;
   }
   printf("\nFinal approximation of e using expression:- %0.7f at x = %d\n", approx,x-1);
   printf("\nThe value of e calculated by the exp function:- %0.7f", exp(1));
   return 0;
}
double approximate(int x, double *approx)
{
   double temp = ((2.0 * x) + 1.0) / ((2.0 * x) - 1.0);
   *approx = pow(temp, x);
}
```

Output -

```
Final approximation of e using expression:- 2.7182828 at x = 476

The value of e calculated by the exp function:- 2.7182818
```