Program 1

Code:

```
#include <stdio.h>

typedef int bool;
#define true 1
#define false 0

int main(void)
{
bool validInput = false;          // Flag for input correctness
bool incorrectEntry = false;      // Flag indicating that the user has
                                  made a mistake in their input
int start = 0;                    // Input start time
int startHours = 0;               // Start time's hours
int startMins = 0;                // Start time's minutes
int duration = 0;                 // Input duration length
int durationHours = 0;            // Duration's hours
int durationMins = 0;             // Duration's minutes
int endTime = 0;                  // Output end time
int endHours = 0;                 // End time's hours
int endMins = 0;                  // End time's minutes

// This loop loops until a valid start time has been entered.
while (!validInput)
{
      // This test changes the prompt message to reaffirm that the user
      has made a mistake in their input.
      if (incorrectEntry)
            printf("Please enter a correct start time: ");
      else
            printf("Please enter the start time: ");

      scanf("%d", &start);

      // Checks to make sure that the start time is not negative.
```

```
      if (start < 0)
      {
            printf("Start time cannot be negative. \n");
            incorrectEntry = true;
            continue;
      }

      startHours = start / 100;   // Hours begin in the hundreads
                                  column.
      startMins = start % 100;    // Minute values end in the tens
                                  column.

      // Checks to make sure that the start time is less than a day.
      if (startHours > 23)
      {
      printf("Start time must be less than a day. \n");
      incorrectEntry = true;
      continue;
      }

      // Checks to make sure that the start time is in the proper
      format.
      if (startMins > 59)
      {
            printf("Start time must be in 24-hour format. \n");
            incorrectEntry = true;
            continue;
      }
      validInput = true;
}

// Resetting the flags for the next input:
validInput = false;
incorrectEntry = false;
```

```
// This loop loops until a valid duration has been entered.
while (!validInput)
{
      // This test changes the prompt message to reaffirm that the user
      has made a mistake in their input.
      if (incorrectEntry)
            printf("Please enter a correct duration: ");
      else
            printf("Please enter the duration: ");

      scanf("%d", &duration);

      durationHours = duration / 100;  // Hours begin in the hundreads
                                       column.
      durationMins = duration % 100;   // Minutes end in the tens
                                       column.

      // Checks to make sure that the duration is in the proper format.
      if (durationMins > 59 || durationMins < -59)
      {
            printf("Duration must be in 24-hour format. \n");
            incorrectEntry = true;
            continue;
      }

      validInput = true;
}

// Calculating the end time:

endMins = startMins + durationMins;    // endMinutes cannot exceed 118
                                       or deceed -118.
endHours = startHours + durationHours;
```

```
// As minutes are sexagesimal, the ending hours must be adjusted
whenever the ending minutes exceed 60 or deceeds -60.

// For minute values between 59 and 118, hours must be incremented:
if (endMins > 59)
{
      endMins %= 60;
      endHours++;
}

// for minute values less than 0, hours must be decremented.:
else if (endMins < 0)
{
      endMins %= 60;

      // C 89/90 does not adjust the result of a modulo to have the
      same sign as the dividend, thus, it must be done manually:
      if (endMins < 0)
      endMins += 60;

      endHours--;
}

// To make sure we return an hour value that does not exceed a day,
the modulo of endHours must be taken:
endHours %= 24;

// C 89/90 does not adjust the result of a modulo to have the same
sign as the dividend, thus, it must be done manually:
if (endHours < 0)
      endHours += 24;

//Finally, the hours and minutes are combined together to obtain the
end time.
endTime = endHours * 100 + endMins;

printf("The end time would be: %04d \n", endTime);

return 0;
}
```

Cases:

6420 -456

obelix[12]% timeAfterTime

Please enter the start time: 6420

Start time must be less than a day.

Please enter a correct start time:

…

[This will continue until an input under 2400 is used.]

2064 +456

obelix[13]% timeAfterTime

Please enter the start time: 2064

Start time must be in 24-hour format.

Please enter a correct start time:

…

[This will continue until a proper input is used.]

456 +2064

obelix[14]% timeAfterTime

Please enter the start time: 456

Please enter the duration: 2064

Duration must be in 24-hour format.

Please enter a correct duration:

…

[This will continue until a proper input is used.]

456 +500

obelix[15]% timeAfterTime

Please enter the start time: 456

Please enter the duration: 500

The end time would be: 0956

1234 +3750

obelix[16]% timeAfterTime

Please enter the start time: 1234

Please enter the duration: 3750

The end time would be: 0224

1234 -3750

obelix[17]% timeAfterTime

Please enter the start time: 1234

Please enter the duration: -3750

The end time would be: 2244

1234 -1250

obelix[18]% timeAfterTime

Please enter the start time: 1234

Please enter the duration: -1250

The end time would be: 2344

123 +456

obelix[19]% timeAfterTime

Please enter the start time: 123

Please enter the duration: 456

The end time would be: 0619

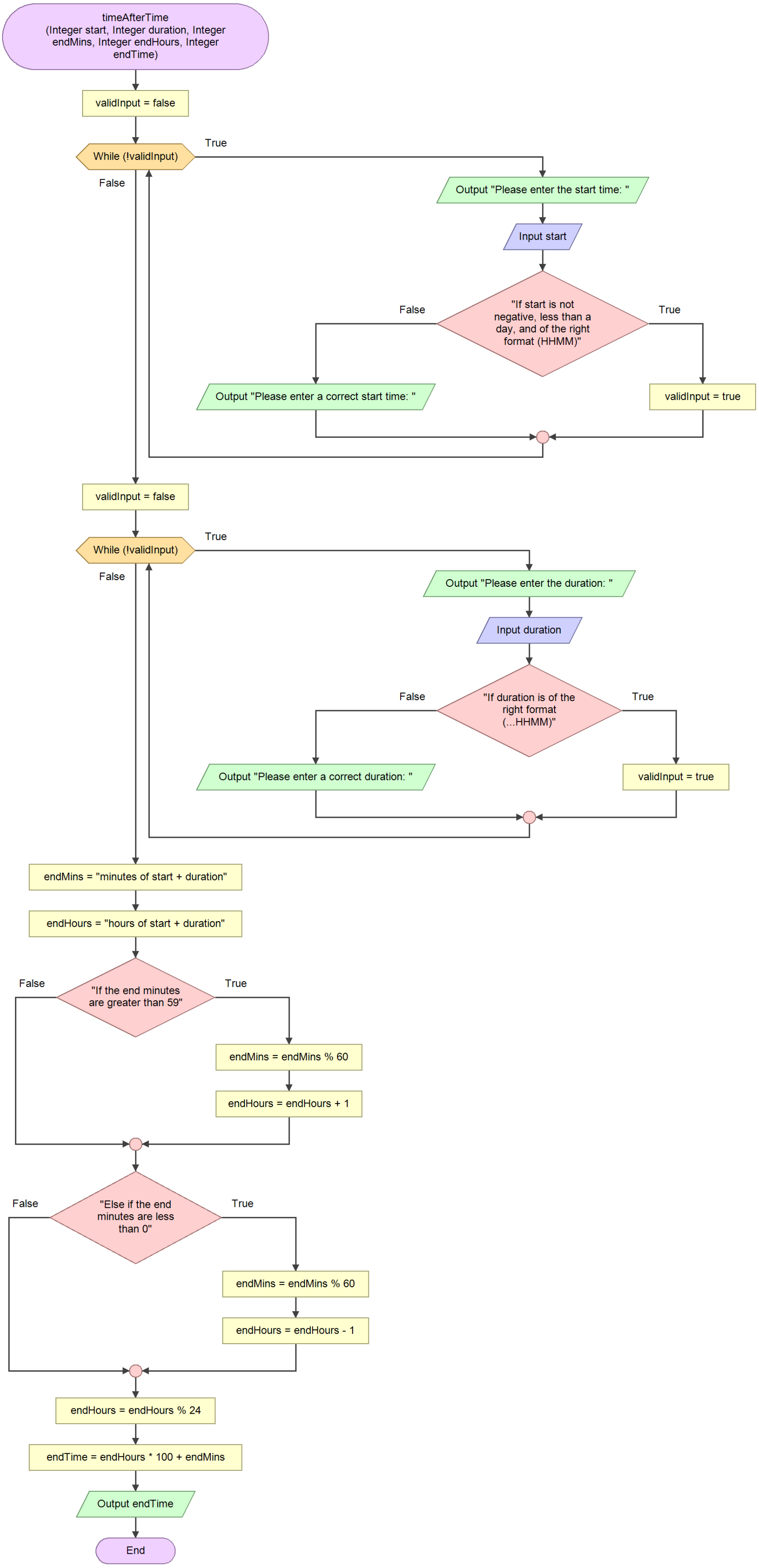
3 +4

obelix[20]% timeAfterTime

Please enter the start time: 3

Please enter the duration: 4

The end time would be: 0007


timeAfterTime
(Integer start, Integer duration, Integer endMins, Integer endHours, Integer endTime)
validInput = false
While (!validInput)
True
False
Output "Please enter the start time: "
Input start
"If start is not negative, less than a day, and of the right format (HHMM)"
False
True
Output "Please enter a correct start time: "
validInput = true
validInput = false
While (!validInput)
True
False
Output "Please enter the duration: "
Input duration
"If duration is of the right format (...HHMM)"
False
True
Output "Please enter a correct duration: "
validInput = true
endMins = "minutes of start + duration"
endHours = "hours of start + duration"
"If the end minutes are greater than 59"
False
True
endMins = endMins % 60
endHours = endHours + 1
"Else if the end minutes are less than 0"
False
True
endMins = endMins % 60
endHours = endHours - 1
endHours = endHours % 24
endTime = endHours * 100 + endMins
Output endTime
End

Problem 2

Code:

```
#include <stdio.h>

typedef int bool;
#define true 1
#define false 0

int main(void)
{
bool validInput = false;          // Flag for input correctness
int n = 0;                        // Number of payments/withdrawls to
                                  make
int numPayment = 0;               // Total number of payments made
float balance = 0;                // Current bank balance
float interest = 0;               // The interest rate on the bank
balance
float payment = 0;                // The withdrawl amount
float finalPayment = 0;           // The value of the final payment

// This loop loops until a valid input is inputted.
while (!validInput)
{
      printf("Please enter (seperated by spaces) the principle amount,
      interest rate, monthly payment, and the number of monthly
      payments: ");

      scanf("%f %f %f %d", &balance, &interest, &payment, &n);

      // Checks to make sure inputs are all positive. Interest rate was
      ommitted to allow this program to use real interest rates, which
      can be negative.
      if (balance < 0 || payment < 0 || n < 0)
      {
            printf("Values must be positive. \n");
            continue;
      }

      validInput = true;
}
```

```
interest /= 100;      // Interest is converted from an integer to a
decimal.

while (n > 0 && balance >= 0)
{
      numPayment++;         // Increment the number of payments.

      // Applies the interest rate and adjust for the withdrawl of the
      payment.
      balance = balance * (1 + interest) - payment;

      // Outputs a message with "payment" used singularly for the first
      payment.
      if (numPayment == 1)
            printf("The balance after %d payment of %.2f would
            be: %.2f. \n", numPayment, payment, balance);

      // As long as the bank account has enough money, keep
      withdrawing.
      else if (balance > 0)
            printf("The balance after %d payments of %.2f would
            be: %.2f. \n", numPayment, payment, balance);

      // Once the bank account has finally run out of money, withdraw
      the last little bit.
      else
      {
            finalPayment = payment + balance;

            printf("The final payment after %d payments of %.2f would
            be: %.2f. \n", numPayment, payment, finalPayment);
            break;
      }

      n--;            // Decrement n as it represents the number of
                      payments to make.
}

return 0;
}
```

Cases:

Case 1

obelix[22]% bankBalance

Please enter (seperated by spaces) the principle amount, interest rate, monthly payment, and the number of monthly payments: 12345 12 1234 15

The balance after 1 payment of 1234.00 would be: 11234.45.

The balance after 2 payments of 1234.00 would be: 10112.79.

The balance after 3 payments of 1234.00 would be: 8979.92.

The balance after 4 payments of 1234.00 would be: 7835.72.

The balance after 5 payments of 1234.00 would be: 6680.08.

The balance after 6 payments of 1234.00 would be: 5512.88.

The balance after 7 payments of 1234.00 would be: 4334.01.

The balance after 8 payments of 1234.00 would be: 3143.35.

The balance after 9 payments of 1234.00 would be: 1940.78.

The balance after 10 payments of 1234.00 would be: 726.19.

The final payment after 11 payments of 1234.00 would be: 733.45.

Case 2

obelix[23]% bankBalance

Please enter (seperated by spaces) the principle amount, interest rate, monthly payment, and the number of monthly payments: 12345 12 543.21 15

The balance after 1 payment of 543.21 would be: 11925.24.

The balance after 2 payments of 543.21 would be: 11501.28.

The balance after 3 payments of 543.21 would be: 11073.08.

The balance after 4 payments of 543.21 would be: 10640.61.

The balance after 5 payments of 543.21 would be: 10203.80.

The balance after 6 payments of 543.21 would be: 9762.63.

The balance after 7 payments of 543.21 would be: 9317.05.

The balance after 8 payments of 543.21 would be: 8867.01.

The balance after 9 payments of 543.21 would be: 8412.47.

The balance after 10 payments of 543.21 would be: 7953.38.

The balance after 11 payments of 543.21 would be: 7489.70.

The balance after 12 payments of 543.21 would be: 7021.39.

The balance after 13 payments of 543.21 would be: 6548.40.

The balance after 14 payments of 543.21 would be: 6070.67.

The balance after 15 payments of 543.21 would be: 5588.17.

Case 3

obelix[24]% bankBalance

Please enter (seperated by spaces) the principle amount, interest rate, monthly payment, and the number of monthly payments: 54321 12 543.21 15

The balance after 1 payment of 543.21 would be: 54321.00.

The balance after 2 payments of 543.21 would be: 54321.00.

The balance after 3 payments of 543.21 would be: 54321.00.

The balance after 4 payments of 543.21 would be: 54321.00.

The balance after 5 payments of 543.21 would be: 54321.00.

The balance after 6 payments of 543.21 would be: 54321.00.

The balance after 7 payments of 543.21 would be: 54321.00.

The balance after 8 payments of 543.21 would be: 54321.00.

The balance after 9 payments of 543.21 would be: 54321.00.

The balance after 10 payments of 543.21 would be: 54321.00.

The balance after 11 payments of 543.21 would be: 54321.00.

The balance after 12 payments of 543.21 would be: 54321.00.

The balance after 13 payments of 543.21 would be: 54321.00.

The balance after 14 payments of 543.21 would be: 54321.00.

The balance after 15 payments of 543.21 would be: 54321.00.

Case 4

obelix[25]% bankBalance

Please enter (seperated by spaces) the principle amount, interest rate, monthly payment, and the number of monthly payments: 54321 12 321 15

The balance after 1 payment of 321.00 would be: 54543.21.

The balance after 2 payments of 321.00 would be: 54767.64.

The balance after 3 payments of 321.00 would be: 54994.32.

The balance after 4 payments of 321.00 would be: 55223.26.

The balance after 5 payments of 321.00 would be: 55454.49.

The balance after 6 payments of 321.00 would be: 55688.03.

The balance after 7 payments of 321.00 would be: 55923.91.

The balance after 8 payments of 321.00 would be: 56162.15.

The balance after 9 payments of 321.00 would be: 56402.77.

The balance after 10 payments of 321.00 would be: 56645.80.

The balance after 11 payments of 321.00 would be: 56891.25.

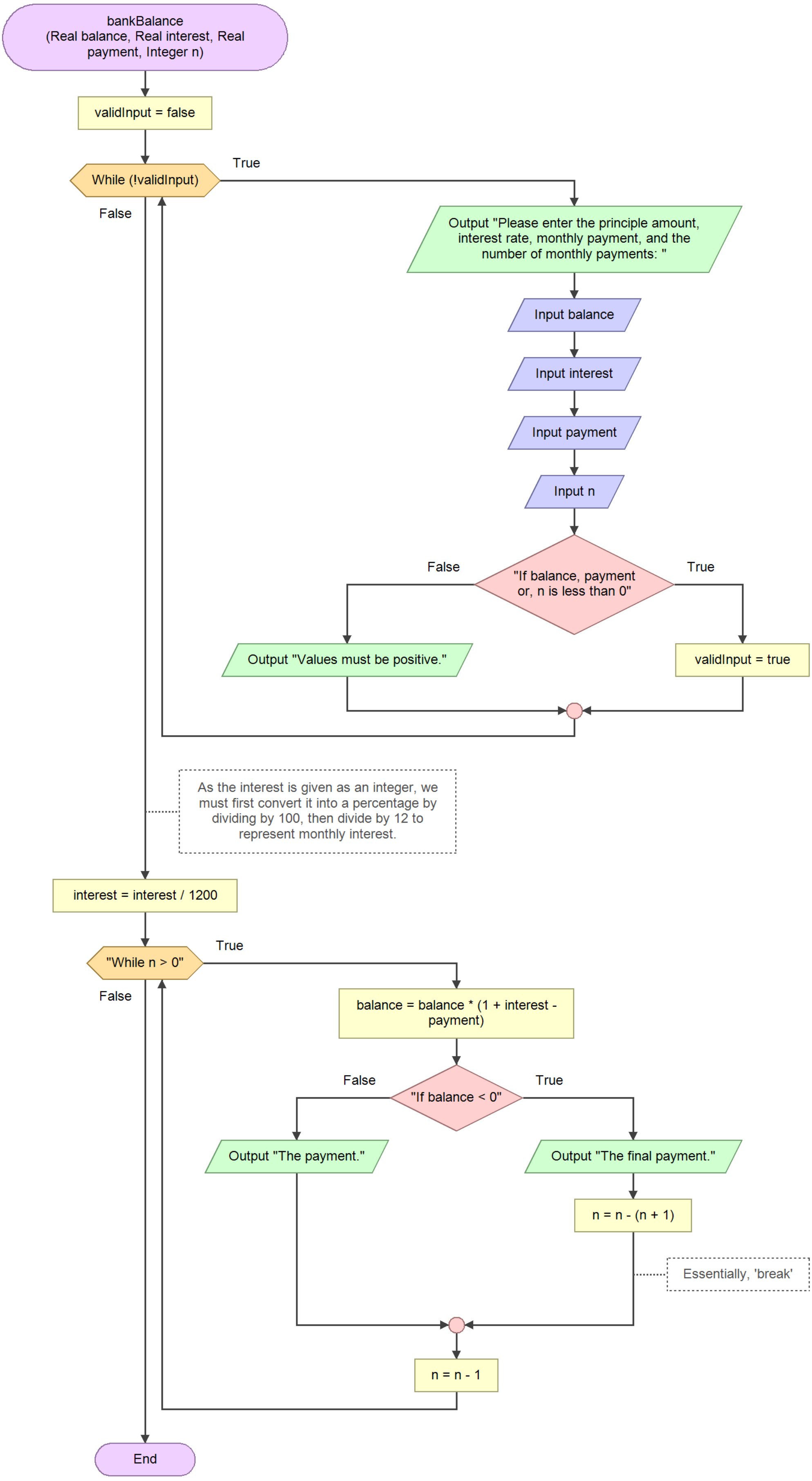
The balance after 12 payments of 321.00 would be: 57139.16.

The balance after 13 payments of 321.00 would be: 57389.55.

The balance after 14 payments of 321.00 would be: 57642.45.

The balance after 15 payments of 321.00 would be: 57897.88.

For the sake of this flowchart, consider "Real" as float.



Problem 3

Code:

```
#include <stdio.h>

typedef int bool;
#define true 1
#define false 0

int main(void)
{
bool validInput = false;          // Flag for input correctness
bool smallNumber = false;         // Flag for an input that is too small
int i = 2;                        // The factorial factor, n in n! = n *
(n-1)!
double n = 1;                     // The denominator of a single element
                                  in the Euler approximation formula,
                                  ie. 1 + 1/n!...
double e = 1;                     // The approximation of Euler's
                                  constant
double fact = 0;                  // The value of a single element in
                                  the Euler approximation formula
double input = 0;                 // The input value

// This loop loops until a valid input has been inputted.
while (!validInput)
{
      // If the user entered a number that was not sufficiently small,
      they are prompted to enter a smaller number.
      // Otherwise, they are just prompted for a number.
      if (smallNumber)
            printf("Please enter a smaller, positive, decimal number:
            ");
      else
            printf("Please enter a small, positive, decimal number: ");

      scanf("%lf", &input);
```

```
      // Checks to make sure the input is not negative.
      if (input < 0)
      {
            printf("Number must be positive. \n");
            continue;
      }

      // Checks to make sure that the input is sufficiently small.
      if (input > 1)
      {
            smallNumber = true;
            continue;
      }

      validInput = true;
}

while (true)
{
      n *= i;               // The denominator retains (n-1)! and is
                            multiplied by n to give n!

      fact = 1/n;

      // Checks to see if this element is smaller than the inputted
      value. If it is, stop the approximation.
      if (fact < input)
                  break;


      e += fact;            // Adding the new factorial value to
                            approximate Euler's number.
      i++;                  // Increment i to prepare it for the next
                            factorial.
}

printf("The decimal approximation of Euler's number to the %lgth
is: %0.15lf. \n", input, e);

return 0;
}
```

Cases:

0.01

obelix[26]% approxE

Please enter a small, positive, decimal number: 0.01

The decimal approximation of Euler's number to the 0.01th (after 6 terms) is: 2.708333333333333.

0.001

obelix[27]% approxE

Please enter a small, positive, decimal number: 0.001

The decimal approximation of Euler's number to the 0.001th (after 8 terms) is: 2.718055555555555.

0.0001

obelix[28]% approxE

Please enter a small, positive, decimal number: 0.0001

The decimal approximation of Euler's number to the 0.0001th (after 9 terms) is: 2.718253968253968.

0.00001

obelix[29]% approxE

Please enter a small, positive, decimal number: 0.00001

The decimal approximation of Euler's number to the 1e-05th (after 10 terms) is: 2.718278769841270.

0.000001

obelix[30]% approxE

Please enter a small, positive, decimal number: 0.000001

The decimal approximation of Euler's number to the 1e-06th (after 11 terms) is: 2.718281525573192.

0.0000001

obelix[31]% approxE

Please enter a small, positive, decimal number: 0.0000001

The decimal approximation of Euler's number to the 1e-07th (after 12 terms) is: 2.718281801146385.

0.00000001

obelix[32]% approxE

Please enter a small, positive, decimal number: 0.00000001

The decimal approximation of Euler's number to the 1e-08th (after 13 terms) is: 2.718281826198493.

0.000000001

obelix[33]% approxE

Please enter a small, positive, decimal number: 0.000000001

The decimal approximation of Euler's number to the 1e-09th (after 14 terms) is: 2.718281828286169.

0.0000000001

obelix[34]% approxE

Please enter a small, positive, decimal number: 0.0000000001

The decimal approximation of Euler's number to the 1e-10th (after 15 terms) is: 2.718281828446759.

0.00000000001

obelix[35]% approxE

Please enter a small, positive, decimal number: 0.00000000001

The decimal approximation of Euler's number to the 1e-11th (after 16 terms) is: 2.718281828458230.

For the sake of this flowchart,
consider "Real" as double.