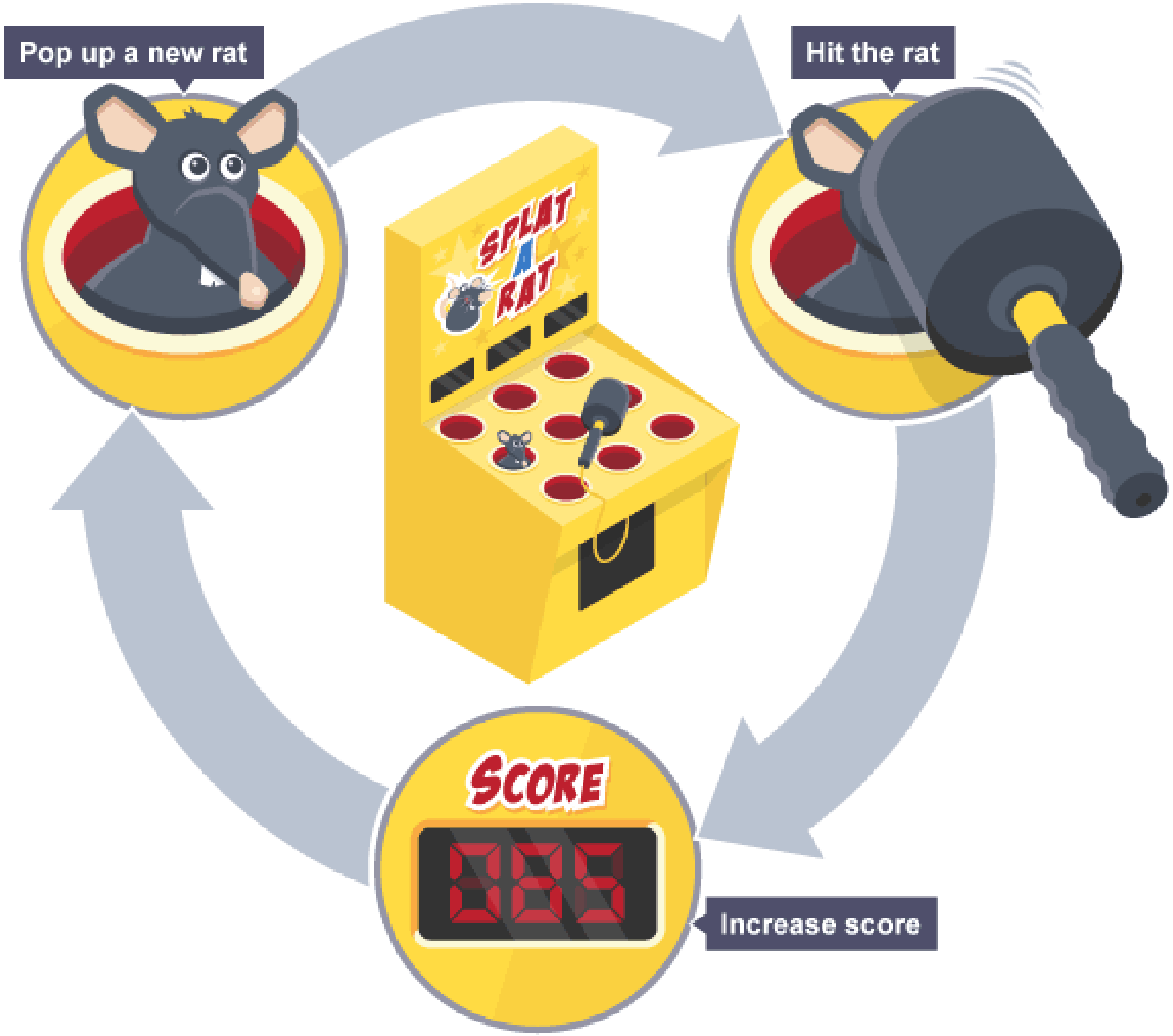

Pop up a new rat
Hit the rat
SPLAT A RAT
SCORE
085
Increase score

Outline chapter 4
control structures: REPETITION
READ/STUDY SEC 4.1 TO 4.6

STUDY AND PRACTICE WITH CODING NEWBIE VIDEO

12 FOR loops

13 WHILE loops

14 DO WHILE loops

in this order: The text covers While loop first.

Objectives

1. Algorithm Development (FLOWCHARTS) for repetition
2. Repetition Structures "while" "do/while" "for"
3. Special use of C++ `break` and `continue` Statements
4. Study of text applied problems

   GPS Data & Weather Balloons

The `while` Statement

Syntax:

```
while (boolean_expression)
        statement;
```

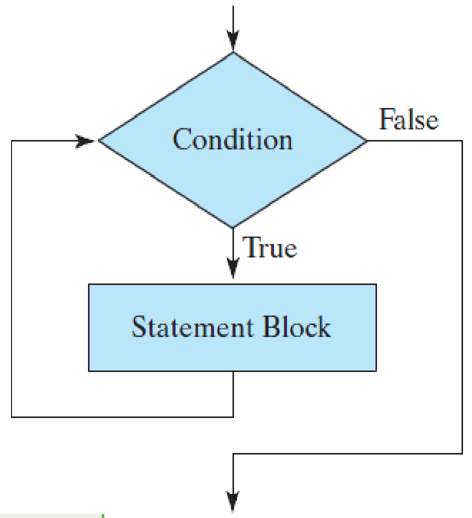
```
while (boolean_expression)
 {
          statement1;
              …
          statement_n;
 }
```

Examples

```
while (!isspace(ch))
{
     cin.get(ch);  //statement repeated until a
}                  // space character is read

                     (‘ ‘  ’/t’  ’/n’ etc)

While LOOP
x=10;
y=3;
while (x>y) {  //statement block executed
               //while x>0
     ++cl;     // “cl” gets incremented
     --x;    // loop variable changes
}
```



Infinite loops used to crash computers

- An infinite loop is one in which the loop condition is always true.
- This can happen if you do not change the control variable HERE IT IS “t” in
- The block of code under the while loop
- for example
- t=1
- While (t<100)
- {
- …
- …
- …
- t=t+1   YOU MUST HAVE A CHANGE IN THE CONTROL VARIABLE USUALY THE LAST STATEMENT IN A WHILE LOOP!
- }
- What happens if the ++ t   is left out of the block?

Avoiding Bugs

- When programming longer programs,
  it is not uncommon to have a large number of error messages from the compiler.
  - A single error often causes several errors as the compiler gets ‘confused’ over what follows an error.
  - Start correcting errors from the top of the error list, NOT the bottom.
  - Recompile your program frequently, limiting the places where errors may occur.
- Use endl in debugging output statements to be sure what you send is actually seen on screen.

```
/* IN CLASS DEMO                                          */
/* This program prints a degree-to-radian table                */
/* using a while loop structure.                               */
#include<iostream> //Required for cout
#include<iomanip> //Required for setw()
using namespace std;
const double PI = 3.141593;
int main()
{
  // Declare and initialize objects.
  int degrees(0);
  double radians;
  // Set formats.
  cout.setf(ios::fixed);
  cout.precision(6);
  // Print radians and degrees in a loop.
  cout << "Degrees to Radians \n";
  while (degrees <= 360)
  {
    radians = degrees*PI/180;
    cout << setw(6) << degrees << setw(10) << radians << endl;
    degrees += 10; // remove for infinite loop demo
  }// end loop
  // Exit program.
  return 0;
} //end main
```

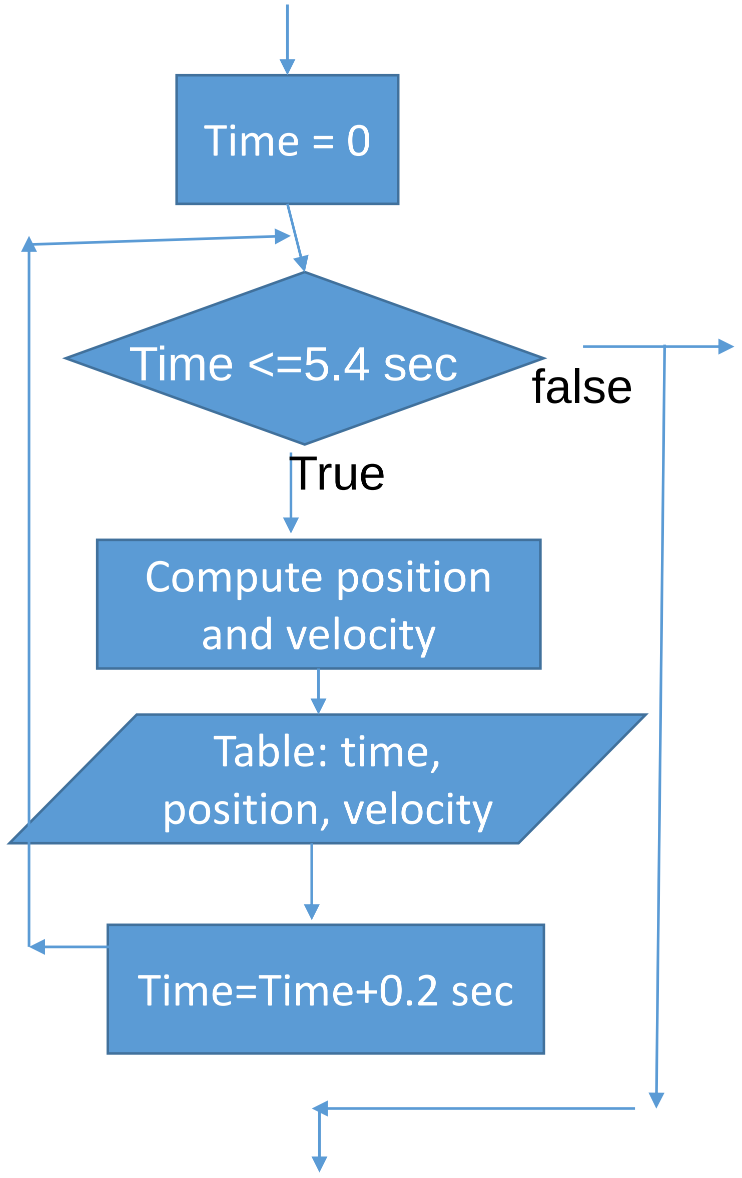
Motion with constant acceleration

- v = final velocity
- v_0 = initial velocity
- a= acceleration (the change in velocity per time)
- $x=x_0 + v_0 t + 1/2at^2$
- $v = v_0 + at$

- Also note that all objects fall at the same rate as discovered by
- Galileo a= gravity constant = 9.80 m/s an amazing fact!

LAB #14 HAND IN LABORATORY TASK: Flow chart Translation

Computing position and velocity over a Range of Time


Time = 0
Time <=5.4 sec
false
True
Compute position and velocity
Table: time, position, velocity
Time=Time+0.2 sec

WRITE THE PROGRAM FROM THIS FLOW CHART on the left **Not from the text**

- Note: Use While loop
- t=time(sec) x= position(m) v=velocity(m/s)
- v_0 = initial velocity, a=constant acceleration(m/s^2)
- $v=v_0 +at$ $x= v_0 t +1/2at^2$ (note at $t=0, x_0=0$)
- Output **neat table** for t , x and v! (headings!)
- Headings have units. Example for here only
- Time(sec) Position (m) Velocity (m/s)
- RUN IT AND HAND IN!
- USE descriptive variables not ‘t’, ’x’ and ‘v’ or ‘a’ add comments for you and user.
- Use v_0 = 5m/s and a=1.8 m/s^2 as known.
- HAND CALCULATION FOR TIME=5.0 AS CHECK-ON EXERCISE
- **Extra credit.** Design with input “v_0 “ and “a”
- RUN FOR AT LEAST TWO VALUES OF THESE

The do-while Statement

Syntax:

```
do
        statement;
while (boolean_expression);
```

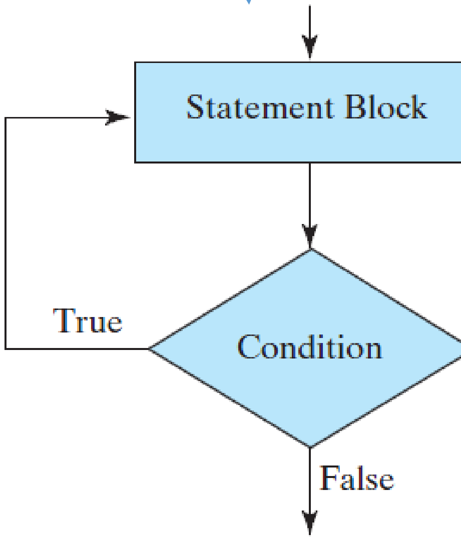
```
Do
{
          statement1;
               …          // block
          statement_n;
}
while (boolean_expression);
```

Examples

```
do
     cin.get(ch); //statement repeated until an
while (ch!=‘\n’);   // newline character is read

do {  //statement block executed (AT LEAST ONCE
     v = a*t + v0;
     cout << t << “ “ << v << endl;
     t += 0.5;    (NOTE LOOP VARIBLE MUST CHANGE)
} while (t<10); //while t< 10
```

Focus on this



In class Exercise: write down the output of the following program segments

A.
```
int d(10); // show variable values in table for this case
   double r;
    do
    {
       r=d*3.14/180
       cout << setw(6) << d << setw(6) <<r<< endl;
       d=d+10;
     } while ( d<=30);
```

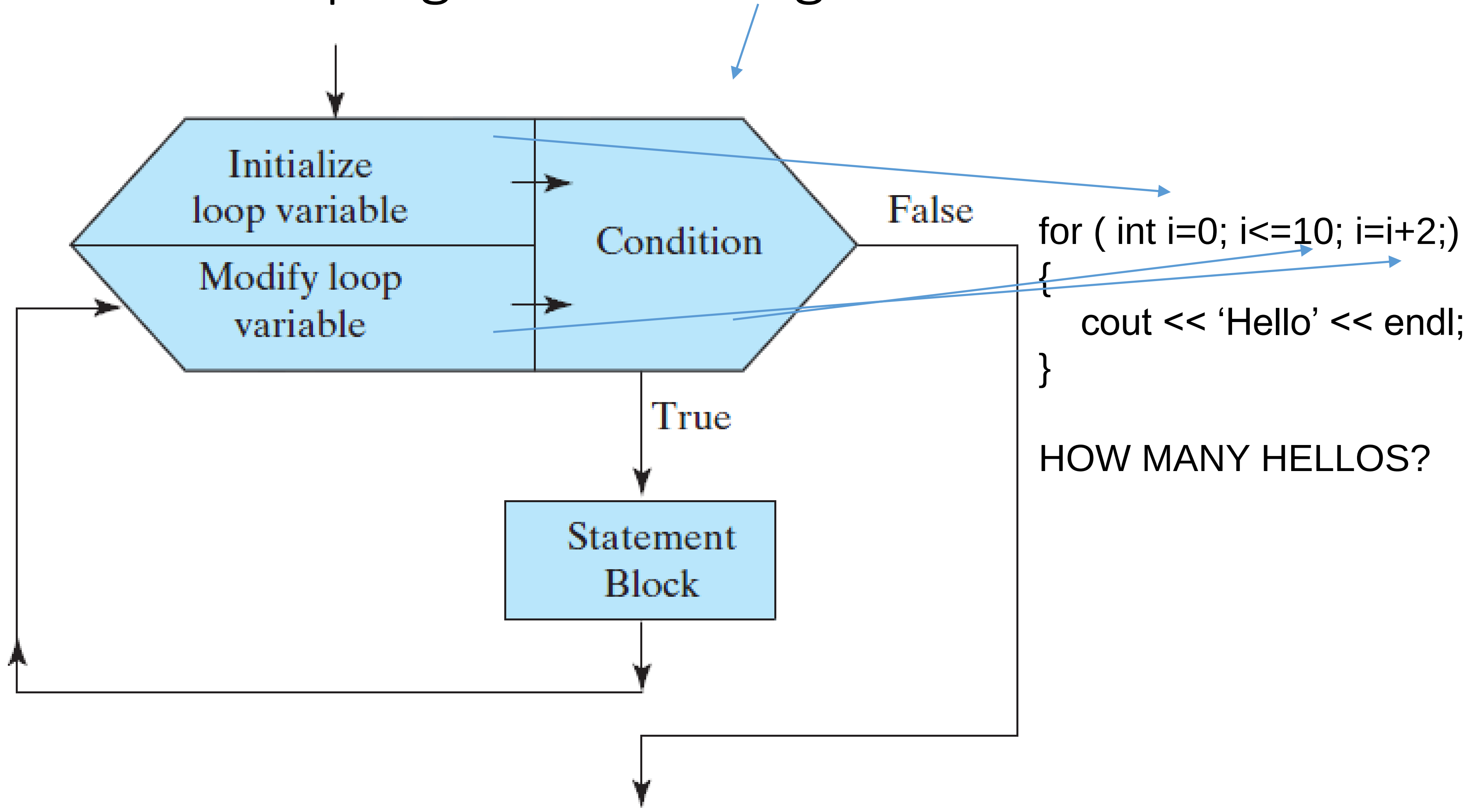
B.
```
int c(1);
    while (c<3)
     {
         ++c
          cout << c<< endl;
     }
```

C.
```
int count(0);
    do
    {
       count = count +3;
     }   while  (count >=9)
     cout << count << endl;
```

D.
```
int c(0);
    do
     {
        c=c+3;
      } while (c>=11);
       cout << c<< endl;
```

FOR LOOP The `for` Statement

note special flowchart symbol
a full control on loop -very useful
in scientific programs . The figure is called?



```
for ( int i=0; i<=10; i=i+2;)
{
    cout << ‘Hello’ << endl;
}
```

HOW MANY HELLOS?

More on the `for` Statement

Syntax:

```
for (initialization_stmt; boolean_condition;  modification_stmt)
{
        statement(s);   // BLOCK
}
```

Examples WHAT IS THE OUTPUT? Play computer(table for variables,i,sum

```
int sum=0;
for (int i = 1; i <= 10; i++)
     sum += i;
cout << sum << endl;
     …
int sum=0;
for (int i = 1; i < 11; ++i)
     sum += i;
cout << sum << endl;
```

For loop possibilities and counting execution times!

How many times does these loops execute?

```
for (int k=1;k<=10; ++k)
  {
     statements;
   }                                          or n-=2

for (n=20; n>0; n=n-2;)
    {
       statements
     }
```

How many time does this one execute? This is for a block

```
   for (int k=5; k< =83; k+=4) { statements }
```

Secret for counting number with <= in the condition

floor ((Final –Initial)/increment + 1)

Hand in HW #8 30 pts read/study 4.1-4.2 label all numbers you did clearly

1. 10 pts Do Chap 4 exam practice questions 7 to 16 both (3rd and 4th editions)

2. 10 pts Show the program trace (Table OF VARIABLES AND OUTPUTS) of the following program segment using the input stream 5 2 -1 10 for N!

```
            int A, B, N;A = 3; B = 6;
            cout << "Enter a sequence of values " ;
            do
{           cin >> N;
                        A = A + N + B;
                        B ++;
            } while (N < 10);
            B *= 2;
            cout << "(N,A,B) = " << "(" << N << ", " << A << ", " << B << ")" <<endl;
```

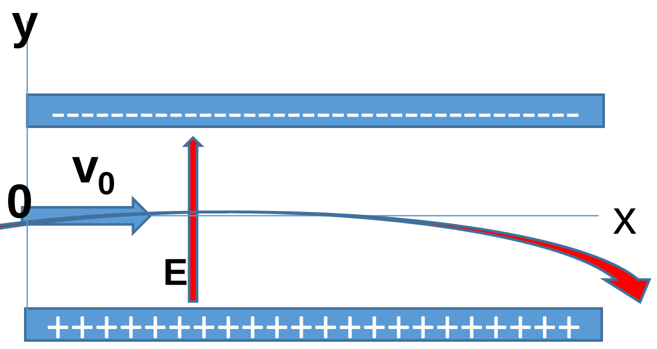
3. 10 PTS Show the program trace (Table OF VARIABLES AND OUTPUTS) for the following program segment: HINT: inner loop starts over each time outer look begins again. This is called nested loops.

```
int I, J;
            for (I=1; I<4; I++)
            {           for (J=4; J>1; J--)
                        {           cout << J << ",  " << I;
                                    cout << endl;
                        }
            }
```

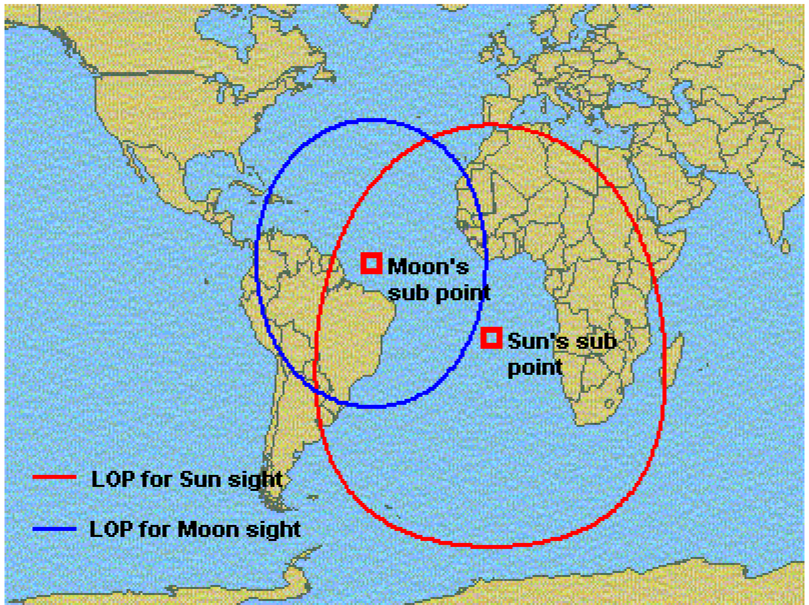
HAND IN LABORATORY TASK: LAB #15
Motion of a charge in an electric field use of the FOR loop

Use **For loop** concept in the following problem.
An electron, charge **e(constant 1.6 x10^{-19} C)** starts out with a velocity $\mathbf{v_0}$ (6 x 10^6 m/s) from position (0,0) (SEE DIAGRAM BELOW) Is moving perpendicular in an electric field **E(2.0x10^{-12} N/C**). The electron will follow a parabolic path given as an **x and y position each of** which can be solved as a function of time **t.** The electron has a mass, **m (constant 9.1 x 10^{-31} kg).**

The electron experiences a force in the **y direction only,** given by **F=eE**
Causing by Newton's second law an acceleration downward =**ma.** Thus the electron will move in the **y** direction with the usual kinematic equation
y =1/2at^2 with "a" being ay constant in the negative direction solved from F=eE=ma The motion in the **x** direction is dependent only on the $\mathbf{v_0}$.
Namely, $\mathbf{x=v_0t}$ **The velocity in the downward direction is v=at**
USE A **FOR** LOOP WITH **t** the loop variable running from **0 to 0.5 sec in 0.01 increments. Create a labeled table of t, x, y, v showing the motional positions, (ie, x and y). Does the x, y and v results make sense? ANSWER. WHY or WHY NOT!**



A study of loops with decision making:
A STUDY OF THE GLOBAL POSTIONING SYSTEM (GPS)
It all started with **Celestial Navigation** .
TYPICALLY A LOCATION CAN BE FOUND AT THE INTERSECTION
 OF THREE CIRCLES ON a flat map of the EARTH SINCE IF YOU
KNOW THE DISTANCE(range) TO THREE REFERENCE POINTS AND
CAN DRAW CIRCLES of proper size ON THE EARTH the Intersection of
 three such circles is our unique position.
The three references on earth are where the star, or moon or Sun is directly over head,
Known As the sub point. Illustrated below are two sub points and known circular sizes
Determined by a sextant instrument that measures the angle to the star or moon or sun.
Better just stars The circle is all The points on the earth that the object observed has
 that angle. Where is the possible location(s) of the observer in this case?
Why is a third circle needed?



Our GPS consideration a little different than celestial
navigation : Since satellites positioning is a three dimensional
picture not a flat map as before we need the intersection of
objects but spheres in 3D to find our position.
The radius of a sphere is the range to the satellite and
the minimum needed is 4 of them for accurate position.

A radio signal carrying data moving at the speed of light is
broadcast to the satellite and rapidly there is received
data. If you know the Time you sent the broadcast, T_b and the time
you received the response T_r , you now know how much the round trip
time called the TRANSIT TIME by taking the difference of the two times.
T_r -T_b and since **Rate x time =distance**
we define the range to the satellite (called the pseudorange,P) as
$P= (T_r - T_b)\, C$
with C the speed of the radio wave which is the speed of light!

Our problem to study is a part of the GPS solution (not all) as follows.

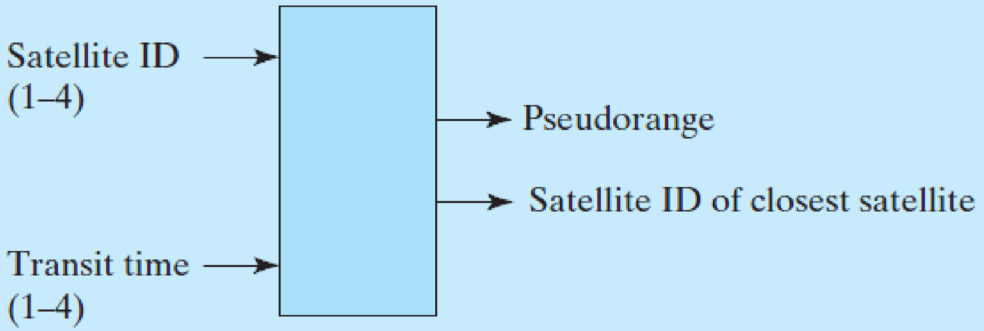
Problem Solving Applied: GPS Data MAXIMA AND MINIMA IN CALCULUS

1. PROBLEM STATEMENT

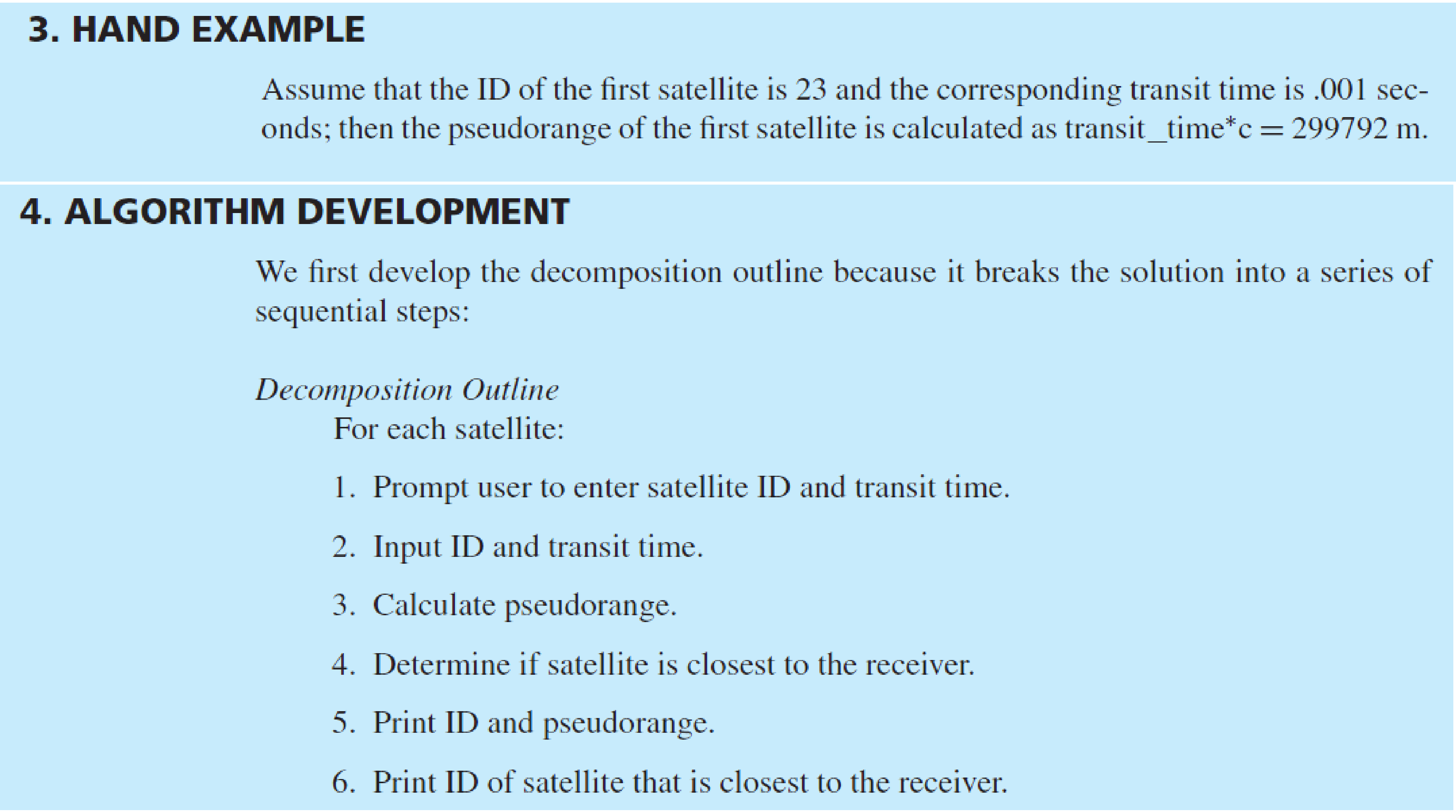
Given the satellite ID and the transit time of broadcasts from four satellites, compute and print the pseudorange for each satellite. Also print the satellite ID code of the satellite that is closest to the GPS receiver.

2. INPUT/OUTPUT DESCRIPTION

The following diagram shows that the input to the program includes four consecutive pairs of data (satellite ID code, transit time). The output is the pseudorange for each satellite followed by the ID of the satellite that is closest to the receiver.



Problem Solving Applied:
GPS Data



3. HAND EXAMPLE

Assume that the ID of the first satellite is 23 and the corresponding transit time is .001 seconds; then the pseudorange of the first satellite is calculated as transit_time*c = 299792 m.

4. ALGORITHM DEVELOPMENT

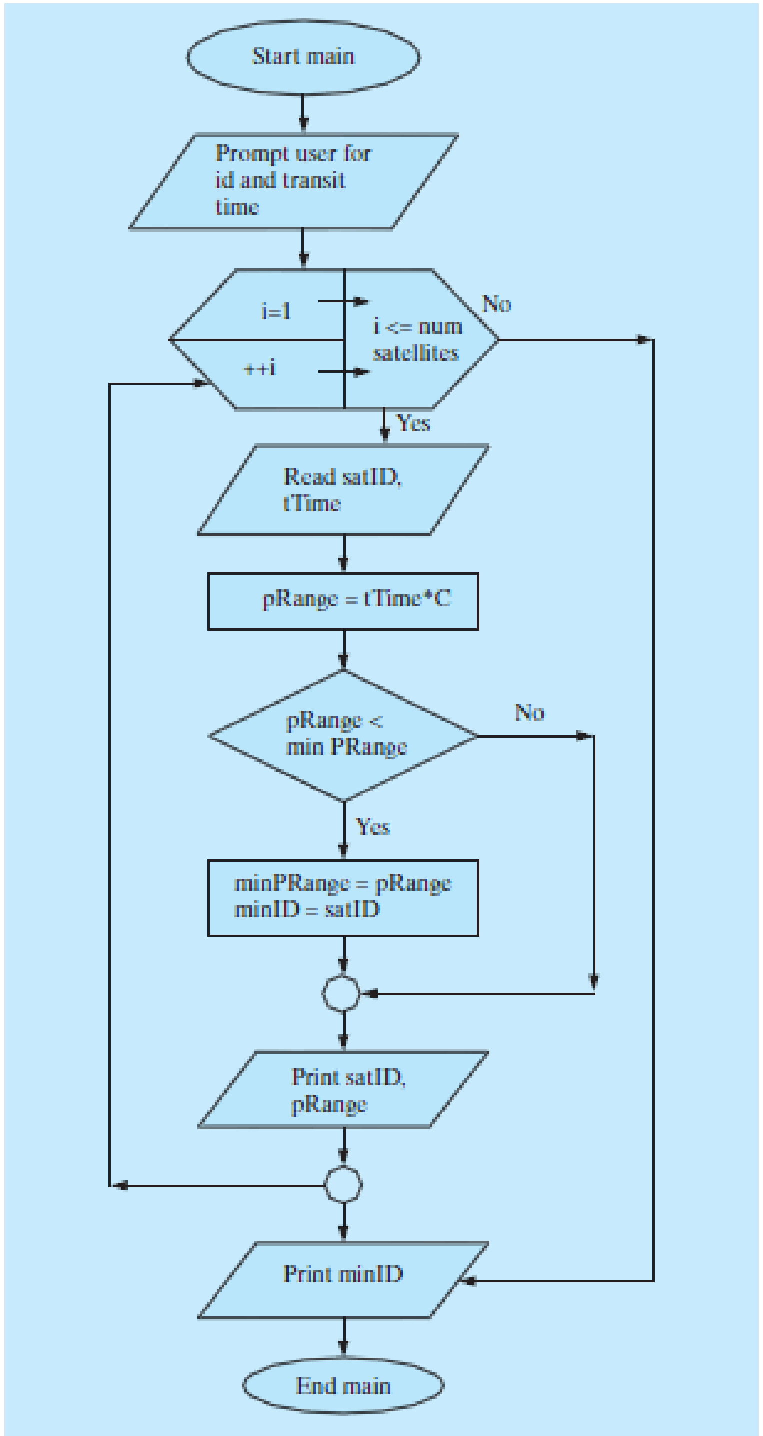
We first develop the decomposition outline because it breaks the solution into a series of sequential steps:

Decomposition Outline

For each satellite:

1. Prompt user to enter satellite ID and transit time.
2. Input ID and transit time.
3. Calculate pseudorange.
4. Determine if satellite is closest to the receiver.
5. Print ID and pseudorange.
6. Print ID of satellite that is closest to the receiver.

Problem Solving Applied: GPS Data



```
// DEMO of finding a minimum or maximum value in a set of values  another FOR loop example
#include<iostream> //Required for cout  prog 4_5!
#include<cfloat>   //required for DBL_MAX
using namespace std;
int main()
{  //Declare and initialize objects
   const double C(299792.458);  //Kilometers per second
   const int NUMBER_OF_SATELLITES(4);
   int satID, minID;
   double tTime, pRange, minPRange(DBL_MAX);
   cout << "Enter id and transit time for " //Prompt user for input << NUMBER_OF_SATELLITES
           << " satellites:\n"
           << "Use whitespace to separate the values(ie: 25 0.00567)\n"
           << "all satellites should be within the lunar orbit of 400,000 km “  << endl;
   for (int i = 1; i <= NUMBER_OF_SATELLITES; ++i)
   {
       cin >> satID >> tTime;
       pRange = tTime*C;
       if (pRange < minPRange) //Check for closest satellite
          {
              minPRange = pRange;
              minID = satID;
           }
       cout << "Satellite " << satID << " has a pseudorange of "
       << pRange << " Km\n" << endl;
   }
   //Output ID of closest satellite
 cout << "\nSatellite " << minID
 << " is closest to GPS receiver." << endl;
 return 0;
}
```

//FINDING MIN AND MAX VALUES VERY
//IMPORANT PROCESS NOTE
TECHNIQUE
 here we set initial minPRrange to max
value
 to find a MAX we set a variable called
 maxPRrange to zero
 and ask   prange>maxPRange

REMINDER EFFICENT CODING COUNTS!

- The next two illustrations show how to efficiently make
- Selections in the logic of your programs.
- It also demonstrates how not to do it and though the poor style was
- Tolerated to now, it no longer will be acceptable programming.
- Keep the latter in mind.   Not sure, ask the professor!

- In each illustration
- Int code;
- cin >> code;

THE FOLLOWING IS VERY BAD PROGRAMMING!
WHY?   Such coding though tolerated by the system your
Engineer boss would find it unacceptable! REMEMBER OUR USE OF “SWITCH”!

```
cin>> code;
If (code=10)
{         cout << “Warning Too hot, turn the system off”<< endl;
}
 if (code= 11)
 {
          cout << “Warning check temperature every 5 minutes ”<< endl;
  }
if (code =13)
  {
           cout << “Warning, better turn on the circulating fan ” << endl;
   }
 if (code=16)
   {
            cout << “Emergency, Abandon the building explosion imminent” << endl;
    }
  else
     {
            cout << “Normal operating range, no problems” << endl;
     {
```

```
If (code=10)     A BETTER and proper way to do the previous (ALSO SWITCH)
        cout << “Warning Too hot, turn the system off”<< endl;
Else
{
      if (code= 11)
                     cout << “Warning check temperature every 5 minutes ”<< endl;
      else
        {
           if (code =13)
                          cout << “Warning, better turn on the circulating fan ” << endl;
           else
                {
                   if (code=16)
                                  cout << “Emergency, Abandon the building explosion imminent” << endl;
                   else
                                  cout << “Normal operating range, no problems” << endl;

                 {
        {       NOTE ONCE ANY CONDITION IS TRUE THE PROGRAM MOVES ON
{      THIS IS EFFICIENT CODING . THE SYSTEM DOES NOT HAVE TO TEST ALL, ALWAYS
```

HAND IN LABORATORY TASK: LAB #16

The program is a modification of the previous satellite program.
Use that code as the basis to do the following
Modify it to find the Satellite that is **farthest and closest** from the GPS receiver using the data Below.
And also modify it for the following 6 satellite data as input. See below
Program will output the satellite number and pseudo range of each and the
And the final output is the farthest and closest satellite from the GPS receiver
Outputting The satellite numbers and pseudo range calculated it found is produced.
Remember to use descriptive variable names and comments to you and the user.
Satellite data to use

Satellite number	measured Transit times
3098	.00234
1243	.00101
1111	.00099
9876	.00567
2018	.00901
7777	.00777

The `break` Statement

- **break;**
  - terminates loop   ANY LOOP IN THIS CHAPTER!
  - execution continues with the first statement following the loop

Example: What is the output?

```
for (int i=0; i<=10; ++i) {
  if(i%2) break;
  cout << i << endl;
}
```

The `continue` Statement

- **continue;**
  - forces next iteration of the loop, skipping any remaining statements in the loop. IE restarts the loop

Example: What is the output?

```
for (int i=0; i<=10; ++i) {
   if(i%2) continue;
   cout << i << endl;
}
```

ILLUSTRATION OF BREAK AND CONTINUE

```
sum =0;
for (int i=1; i<=20; ++i)
 {
  cin >>x;
  if(x>10) break;
  sum = sum + x;
  }
  cout <<sum<< endl;
```

What is sum for if we want inputs 5,7,8,12,9,11

```
sum =0;
for (int i=1; i<=20; ++i)
 {
  cin >>x;
  if(x>10) continue;
  sum = sum + x;
  }
  cout <<sum<< endl;
```

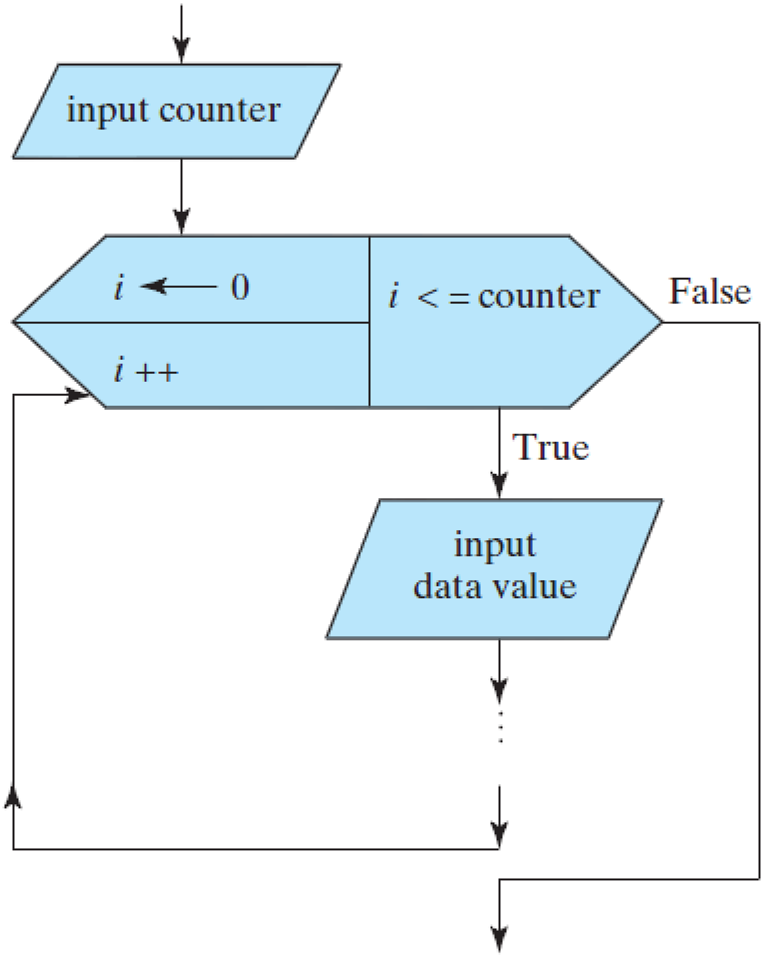
What is sum for if we want inputs 5,7,8,12,9,11

Structuring Input Loops

- **Repetition is useful when inputting data from standard input or from a file.**
- **Common repetition structures:**
  - **counter-controlled –we are Counting!**
  - **sentinel-controlled - special value!**
  - **end-of-data controlled -**

Counter-Controlled Loops

- May be used for reading input data if the number of data values is known before the data are entered.



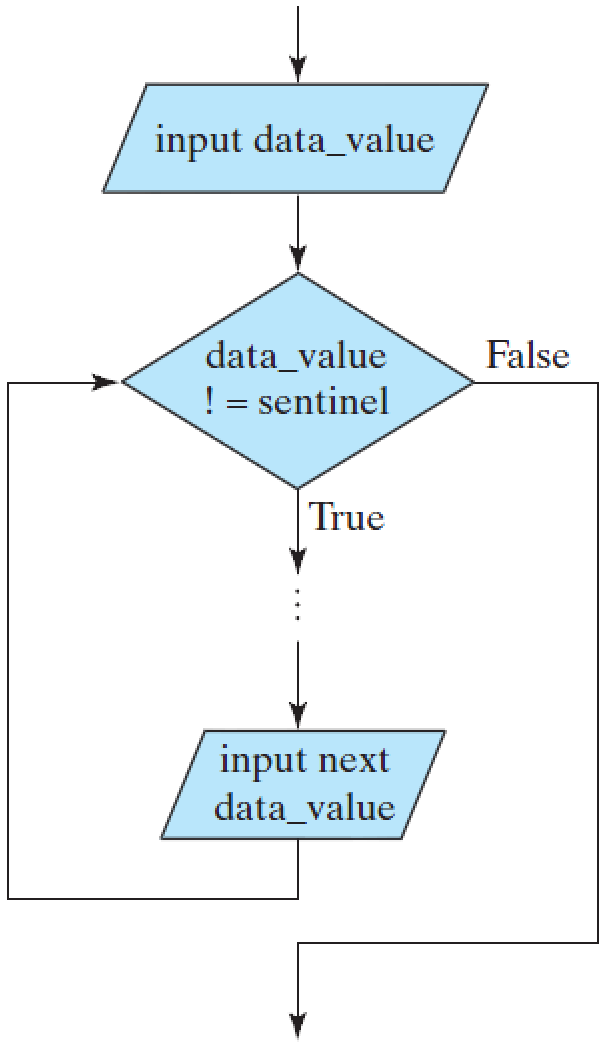
COUNTER CONTROL

```
/* This programs finds the average of a set of exam scores.   */
#include<iostream>//Required for cin, cout
using namespace std;
int main()
{
  double exam_score, sum(0), average; // Declare and initialize objects.
  int counter;
  // Prompt user for input.
  cout << "Enter the number of exam scores to be read ";
  cin >> counter;
  cout << "Enter " << counter << " exam scores separated "
       << " by whitespace ";
  // Input exam scores using counter-controlled loop.
  for(int i=1; i<=counter; ++i)
    {
     cin >> exam_score;
     sum = sum + exam_score;
    }
  // Calculate average exam score.
  average = sum/counter;
  cout << counter << " students took the exam.\n";
  cout << "The exam average is " << average << endl;
  // Exit program
  return 0;
}
```

WHAT IS FINAL OUTPUT FOR 3 10 20 30 ?

Sentinel-Controlled Loops

- May be used for reading input data if a special data value exists that can be used to indicate the end of data.



```
  SENTINEL CONTROL                                          */
/* This programs finds the average of a set of exams */
#include<iostream>  //Required for cin, cout
using namespace std;
int main()
{  // Declare and initialize objects.
  double exam_score, sum(0), average;
  int count(0);
  // Prompt user for input.
  cout << "Enter exam scores separated by whitespace. ";
  // Input exam scores using sentinel loop.
  cin >> exam_score;
  while(exam_score>=0)  what value(s) terminates the loop?
  {
  sum = sum + exam_score;
  ++count;
  cin >> exam_score;
  }
  // Calculate average exam score.
  average = sum/count;
  cout << count << " students took the exam.\n";
  cout << "The exam average is " << average << endl;
  // Exit program
  return 0;
}
```

End-of-data Controlled Loops

- Is the most flexible for reading input data.
- No prior knowledge of the number of data values is required.
- No sentinel value is required.

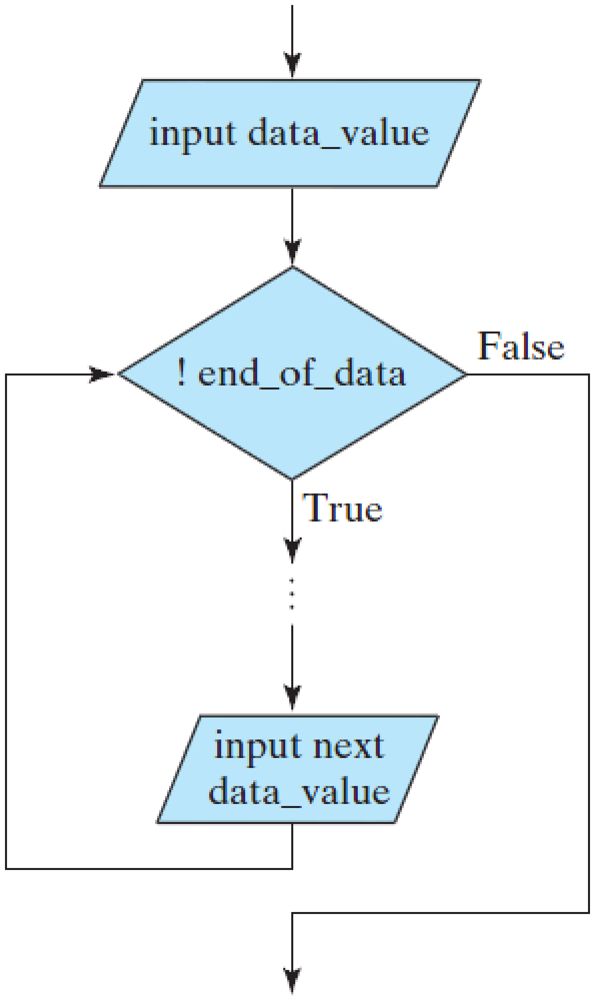
We use a function called “cin.eof()”

returns 0 when special input signally

end of data. This eof =“end of file” used here (also used in files) is system defined.

Key board entry for cin.eof()!

- Use different ctrl + some key? to indicate no more data will be entered
- Our system uses ctrl z anything else will send your output to “computer heaven”!



//End of Data Loop DEMO

```
/* Program chapter4_7                                          */
/* This programs finds the average of a set of exam scores.    */
#include<iostream>  //Required for cin, cout
using namespace std;
int main()
{
  // Declare and initialize objects.
  double exam_score, sum(0), average;
  int count(0);
  // Prompt user for input.
  cout << "Enter exam scores separated by the enter key. ";
  // Input exam scores using end-of-data loop.
  cin >> exam_score;
  while(!cin.eof())            //note use of !
  {
  sum = sum + exam_score;
  ++count;                     //note use of count here.
  cin >> exam_score;
  }
  // Calculate average exam score.
  average = sum/count;
  cout << count << " students took the exam.\n";
  cout << "The exam average is " << average << endl;
  // Exit program
  return 0;
}
```

HW #9 Chap 4 read/study 4.3-4.5 12 pts

1. Do Exam practice questions 1-6 to hand in (both 3rd and 4th ed) 6 pts

Use the following program segment to answer the next two questions

HINT: inner loop starts over each time outer look begins again

```
int I, J;
for (I=2; I>=0; I--)
{       for (J=1; J<4; J++)
                cout << J << ",  " << I << ", ";
        cout << endl;
}
```

2. 3 pts How many times is the statement cout << J << ", " << I << ", "; executed?

A. 0 2 or 3 or 6 or 9 or None of these are correct.

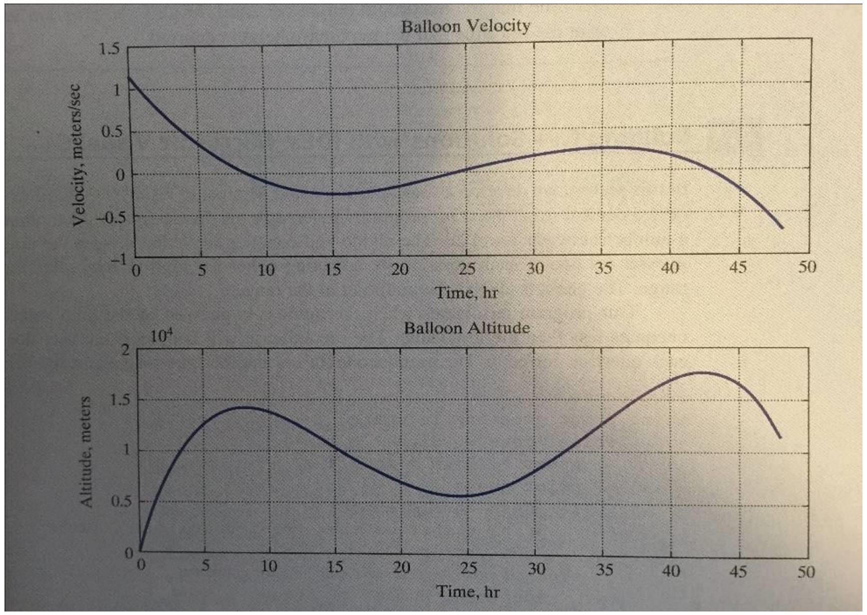
3. 3 pts How many times is the statement cout << endl; executed?

A. 0 2 or 3 or 6 or 9 or None of these are correct.

DEMO Problem Solving Applied: Weather Balloons

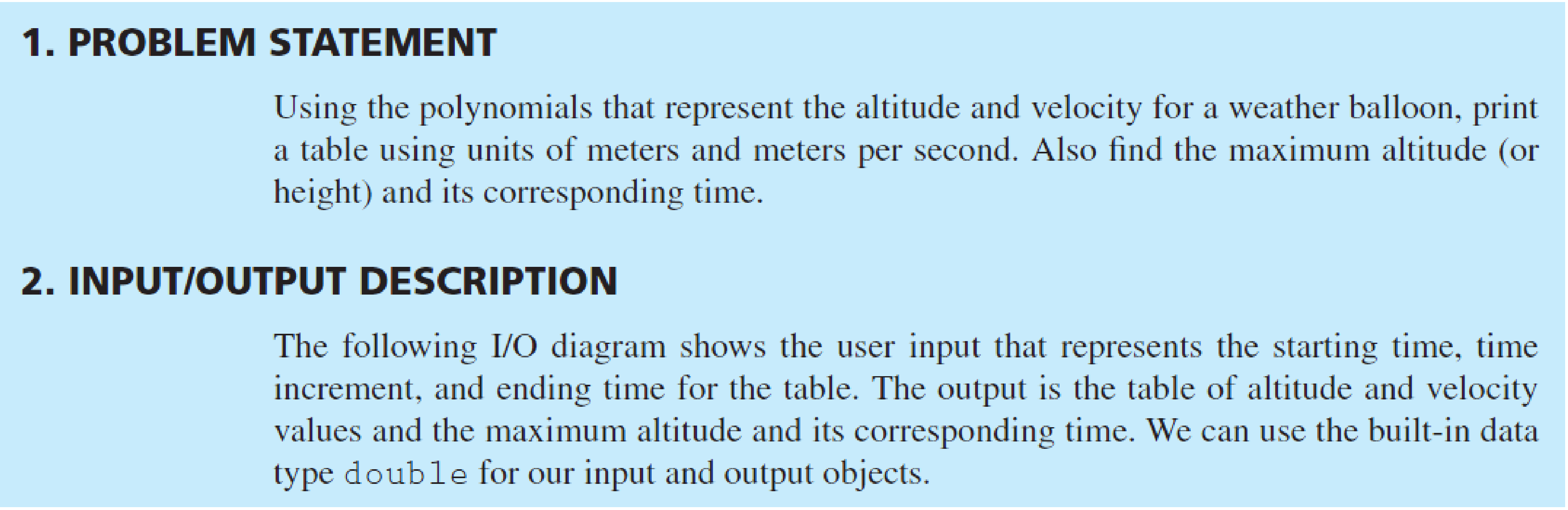
Weather Balloons gather info on the atmosphere as the rise.
Temperature and pressure. They are filled with helium and they rise to a
Specific altitude where the upward buoyant force is balanced by gravity
(equilibrium) but as the air temperature changes during the day and night the
The balloon will expand and contract and rise higher and lower with air
Temperature.. Thus the altitude of the balloon changes as the hours go by.
$alt(t) = -0.12t^4 + 12t^3 - 380t^2 + 4100t + 220$ t in hours alt in meters

The velocity of the balloon is max at the start and as it ascends it will stop at
Its maximum height and then the velocity changes direction as it descends to
a minimum and stop there before rising again, this pattern continues.
$v(t) = -0.48t^3 + 36t^2 - 760t + 4100$ The graph below for 48 hrs shows this situation



later you will learn a simple technique to graph
using a spread sheet

Problem statements: Weather Balloons to construct a flow chart first.



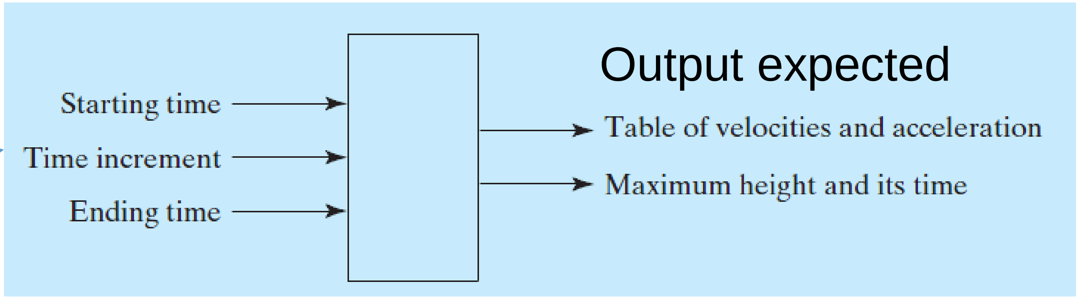
1. PROBLEM STATEMENT

Using the polynomials that represent the altitude and velocity for a weather balloon, print a table using units of meters and meters per second. Also find the maximum altitude (or height) and its corresponding time.

2. INPUT/OUTPUT DESCRIPTION

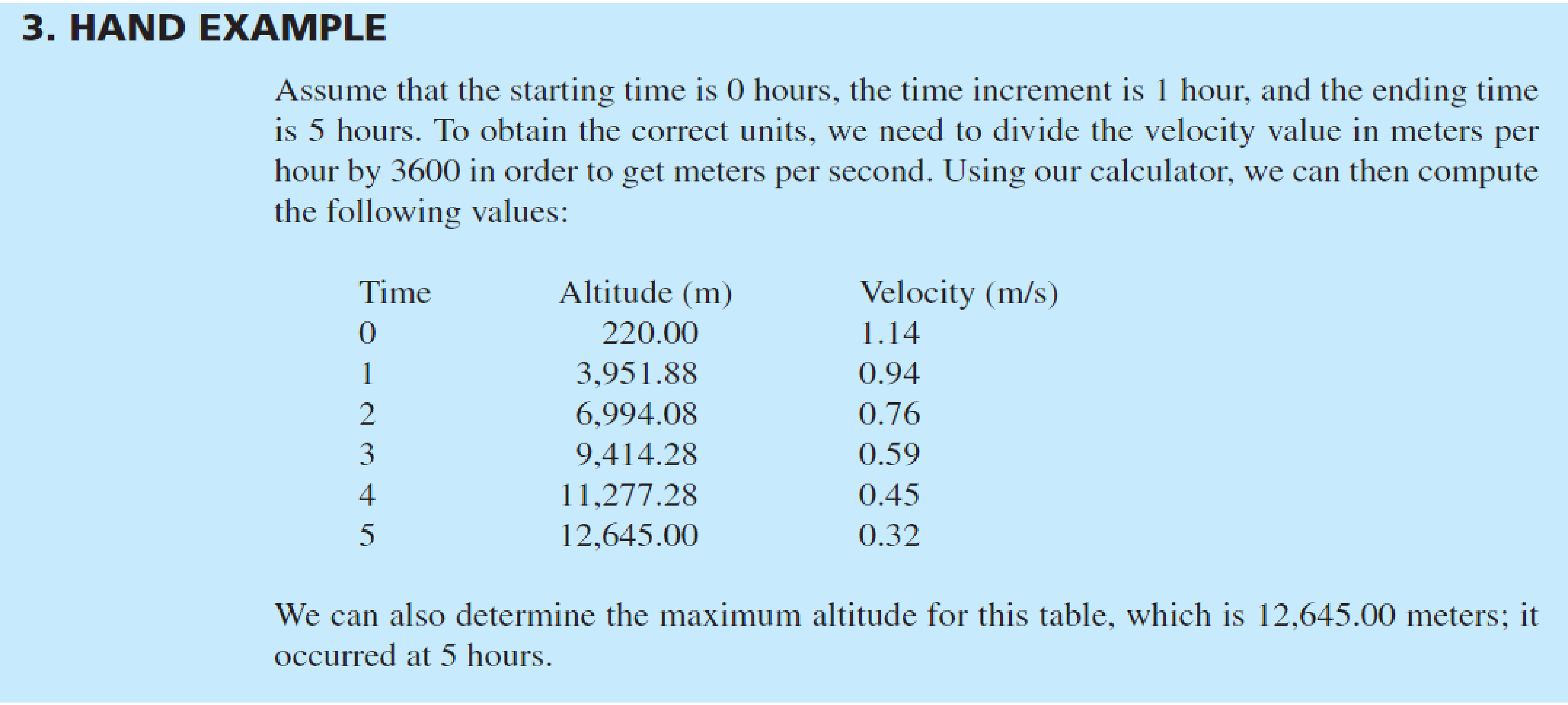
The following I/O diagram shows the user input that represents the starting time, time increment, and ending time for the table. The output is the table of altitude and velocity values and the maximum altitude and its corresponding time. We can use the built-in data type `double` for our input and output objects.

User enters the following 3 items



Starting time
Time increment
Ending time
Output expected
Table of velocities and acceleration
Maximum height and its time

Problem Solving Applied: Weather Balloons: Checking our program!



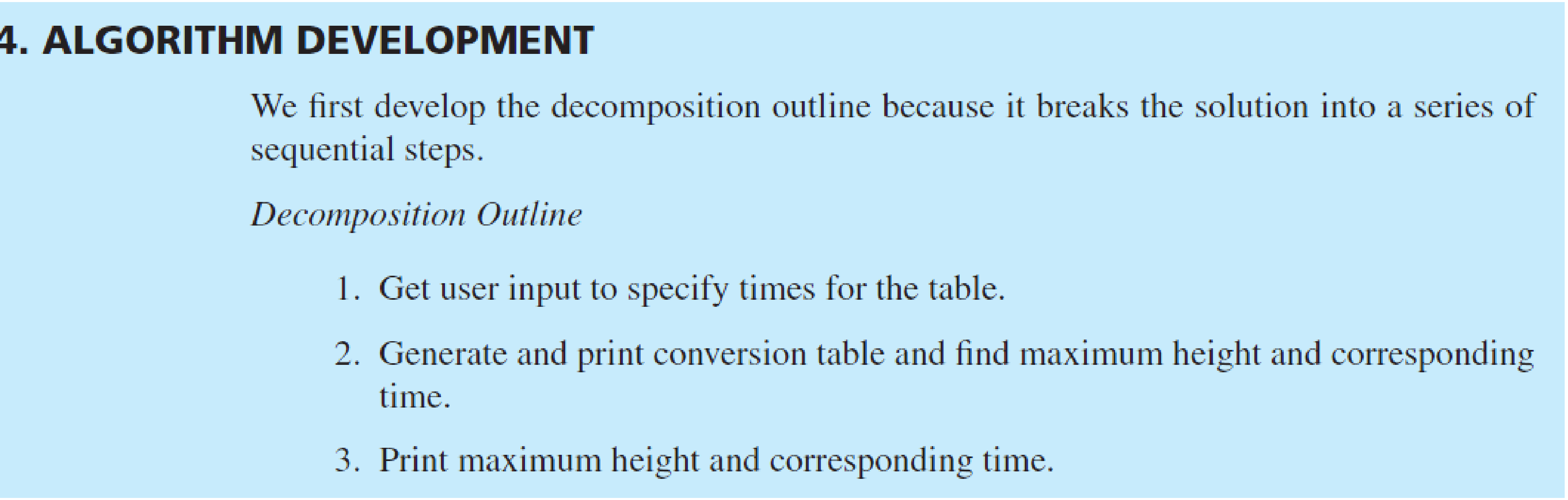
3. HAND EXAMPLE

Assume that the starting time is 0 hours, the time increment is 1 hour, and the ending time is 5 hours. To obtain the correct units, we need to divide the velocity value in meters per hour by 3600 in order to get meters per second. Using our calculator, we can then compute the following values:

Time	Altitude (m)	Velocity (m/s)
0	220.00	1.14
1	3,951.88	0.94
2	6,994.08	0.76
3	9,414.28	0.59
4	11,277.28	0.45
5	12,645.00	0.32

We can also determine the maximum altitude for this table, which is 12,645.00 meters; it occurred at 5 hours.

We want to produce this table and its maximum alt and time to be sure
We coded our program properly before going on to more time data.



4. ALGORITHM DEVELOPMENT

We first develop the decomposition outline because it breaks the solution into a series of sequential steps.

Decomposition Outline

1. Get user input to specify times for the table.
2. Generate and print conversion table and find maximum height and corresponding time.
3. Print maximum height and corresponding time.

I now DEMO the solution from the 5th hour to the 30th hour.

```
/* Program chapter4_8   This program prints a table of height and   velocity values for a weather balloon.              */
#include <iostream>//Required for cin, cout
#include <iomanip>//Required for setw()
#include <cmath>//Required for pow()
using namespace std;
int main()
{
  // Declare and initialize objects.
  double initial, increment, final, time, height,
  velocity, max_time(0), max_height(0);
  int loops;
  // Get user input.
  cout << "Enter initial value for table (in hours) \n";
  cin >> initial;
  cout << "Enter increment between lines (in hours) \n";
  cin >> increment;
  cout << "Enter final value for table (in hours) \n";
  cin >> final;
  // Print report heading.
  cout << "\n\nWeather Balloon Information \n";
  cout << "Time Height Velocity \n";
  cout << "(hrs) (meters) (meters/s) \n";
  // Set formats.
  cout.setf(ios::fixed);
  cout.precision(2);
  // Compute and print report information.
  // Determine number of iterations required.
  // Use integer index to avoid rounding error.
  loops = (int)( (final - initial)/increment );
  for (int count=0; count<=loops; ++count)
  {
    time = initial + count*increment;
    height = -0.12*pow(time,4) + 12*pow(time,3)
             - 380*time*time + 4100*time + 220;
    velocity = -0.48*pow(time,3) + 36*time*time
               - 760*time + 4100;
    cout << setw(6) << time << setw(10) << height
         << setw(10) << velocity/3600 << endl;
    if (height > max_height)
    {
      max_height = height;
      max_time = time;
    }
  }
  // Print maximum height and corresponding time.
  cout << "\nMaximum balloon height was " << setw(8)
       << max_height << " meters \n";
  cout << "and it occurred at " << setw(6) << max_time
       << " hours \n";      /* Exit program. */    return 0;  }
```

Hand in Home work #10 read/study 4.6 35 pts

First study the flow chart in your text for the GPS problem in section 4.3

Note the use of chart diagrams.

1.10 pts Flow chart in detail this program segment

```
cin>> x>>y;
for (int i=1;  i<=20;i++){
       if (x>y)
              w=sqrt(x+y);
       else
               w=log10(x+y);
 }
 cout<< “x= “ <<x<< “ y= “ <<y << “ w= “<<w<<endl;
```

2. 25 pts Now, **Construct a** detailed **flow chart (show variables and actions math and logic in correct flow chart diagrams) of the solution of the Weather balloon Problem use the text solution in chap 4 as your model**

HAND IN LABORATORY TASK: #17(if you get an A grade your earn 20pts for AVG)

FOX HEAVEN an island off the coast of Costa Rica near Isla Sorna

(REPORTS OF STRANGE CREATURES ON THIS ISLAND)

Write a program to study the population of foxes ON Fox Heaven Island as the years go by with the following conditions. **BE SURE TO DO A FLOW CHART FIRST AND HAND IN WITH THE PROGRAM**.
You study the island and count the foxes at 20 with a lot of rabbits, namely 13,000 rabbits.
Looks like plenty of rabbits to keep this island going for a long time. So in your study you find information
About the fox and rabbit reproduction and build this program to predict the future for the island.
As follows:

If every thing is ok, ie. **a GOOD YEAR** foxes can survive by eating 208 rabbits for the year
That means that **a good year is when the total number of rabbits is greater than the present number of foxes times the 208 rabbits(so plenty to eat),** if the present **number of rabbits falls below the good year then it's a BAD YEAR** A AND FOXES WOULD NEED **TO EAT ONLY 52 RABBITS** TO JUST SURVIVE becoming very skinny.

So In a good year the **rabbit population** goes down by number eaten or # foxes *208 and the remaining rabbits goes up by 50%
And the foxes population increases 15% of the previous years value but if year is bad (#of rabbits < # foxes *208) ie. not enough rabbits to fully satisfy the foxes and keep them healthy so they go on a lean diet of only eating 52 rabbits a year as noted above.
So **rabbit population goes Down by #foxes *52** and the **remaining rabbits increase by 50%(they are still healthy).**
The fox population grows in a bad year at only 3% (the fox population is stressed)
These rates of 50% , 15% and 3% includes a combination of births and deaths from various other courses.
THREE MODIFICATIONS HAND IN THE THREE MODIFICATION OUTPUTS WITH THE LAB. With appropriate comments!
1.Write a program with year 1 starting numbers of foxes and rabbits as above. Have it calculate the first good year
And see how many rabbits and foxes there are then check to see if it's a good or bad year and use the rates above.
Output the **year, the fox population, rabbit population** in nice labeled table. Run for 30 years. Can FOX HEAVEN sustain itself, if the rabbit population drops below zero then the foxes all die Set fox number to zero for negative rabbits thus all on the island die. USE A **WHILE LOOP FOR OVERALL RUN AND FOR YEARS** IF-ELSE DECISIONS
Note on your hand in. How many good years, What year(s) did the island turn bad? What year does it die?

2. Take into account that the island has been discovered by "Brits" who hunt foxes in a good year only for good pelts.
The hunters kill 5% of the foxes during the good year **after they reproduce**!. Rerun the program for as many years you need to see if the island Survives, flourishes or dies. If it goes bad what year(s)? And what year does it die if it does? Look at the behavior carefully, anything Interesting? Did the hunters make the situation on the island worse or better or the same in this case? Explain ON OUTPUT
3. Rerun and change % kill of the hunters, figure out how high must it be to keep the island going for 30 year? does Hunting help?