CSCI 4010.001
Design Document
Web Activity Analysis System
Written by: Kyle Davis
March 11, 1999
Hierarchy Chart

Main

Read-and-Write-Options  Get-Field-Values  User-ID is Valid  Report-Generation  Store-Log

getValue  GetThemFields

GetThemData

Total-Hits  Total-Bytes-Received  Top-Domains  Files-Referenced  Daily-Hits  Daily-Bytes  Hourly-Hits  Hourly-Bytes  Hourly-Bytes-Graph  Failures

Please note that this is only right right-side page of the Hierarchy Chart. The student’s document included additional material, but you can get the idea based on what is shown here.
Data Structures

Enumerated type:
The time period will be represented by an enumerated type with the values being:

Today = 0
Yesterday = 1
Last Week = 2
Last Month = 3

An enumerated type will also be used for the determining the order in which reports are generated. The values for the reports will be:

TOTAL_HITS = 1
TOTAL_BYTES_TRANSFERRED = 2
TOP_DOMAINS = 3
FILES_REFERENCED = 4
DAILY_HITS = 5
DAILY_HITS_GRAPH = 6
DAILY_BYTES = 7
DAILY_BYTES_GRAPH = 8
HOURLY_HITS = 9
HOURLY_HITS_GRAPH = 10
HOURLY_BYTES = 11
HOURLY_BYTES_GRAPH = 12
FAILURES = 13

Global variables:
An array of boolean values (checks[]) will be used to determine what options are selected by the user. This array will be used for determining what data needs to be extracted from the log entry. This array will also be used for determining which reports are generated in the final phase.

A global variable will be used for the Fields class pointer in the Web Utility files. This pointer will be used throughout the execution of the cgi script to access information that the user entered in the fields of the form.

Storage Types:
Boolean array mentioned previously (checks[]) will be used to determine which options the user has selected

Enumerated type Report array Order[] will be used when generating the reports to determine the order in which reports are generated

Dynamic array of pointers to structs will be used for the IP addresses
Dynamic array of pointers to structs will be used for the files referenced addresses

Dynamic array of pointers to structs will be used for the failures that occurred

The struct for the above mentioned arrays will contain a string variable and an integer variable which will store a count for this particular string

Boolean variables will be used for graph reports. If a graph is to be generated for the hourly hits, hourly bytes, daily hits, or daily bytes, then a variable for each of these will hold the boolean value true.

An array will be set up for hourly breakdown counters if this option is checked. This array will contain 24 elements that are initialized to 0 and the corresponding element will be incremented when an entry is scanned.

An array will be set up for the daily breakdown counters if this option is checked. Depending upon the time period, the array will be set up like this:

Today - array of 1 element

Yesterday - array of 1 element

Last Week - array of 7 elements

Last Month - array of 31 elements
Pseudo Code

Note: Since this is pseudo code, I do not necessarily specify full variable names in each case.

Main Program(Arguments on command line)
{
    // Command Line
    If this was called from a UNIX environment such as command line then
        Output message like "This software cannot be ran directly from a UNIX command
        line. In order to run the software properly, go to http://m01.csci.unt.edu/~kdavis.
        This software is copyrighted by Kyle Davis 1999."
        Exit

    // Get
    If this was called with "Get" command then
        Call Read_and_Write_Options()
        Exit

    // Post
    If this was called with "Post" command then
        Call Get_Field_Values()
        Get the userid from field 0
        If User_ID_is_Valid(userid) then
            // The following lines are used to decide where to store the logfile that we
            // containing the time and the userid requested
            Open INI file
            Parse the INI file until section [Configuration] is found
            Scan text until you match the string "Log Directory="
            Read and store string after "=" to string variable "dir"
            Call Store_Log(dir, userid)
            Call Report_Generation()
            Close INI file
        else
            Output message to user saying that they entered a username that cannot be
            found on this server
            Exit

    }

Function Read_and_Write_Options()
{
    // parse each line of the INI file and withdraw all information for front page layout
    Open INI file
// [Options Screen] section
While within the options screen section
    Read HTML format text from file
    Output text to browser
    Increment to next line

Close INI file
Return to main program
}

Function Get_Field_Values()
{
    If getValue(0) is equal to 0  // No userid was entered
        Output message to user saying that they never entered a userid to look for
        Exit

    Set up integer variable “array_size”
    Set array_size equal to the value returned from Web Util function GetNumFields()
        -1
        // I make this minus 1 because I don’t want to set up a boolean value for the name
        field

    // Each element in this array will represent one of the fields in the form
    // These values will be used to determine which options are selected
    Create boolean checks[] containing “array_size” elements
    Initialize all elements to false

    For i = 1 to array_size -1 do  // skip over name field
        If Web Util function getValue(i) of element is checked then
            Place “true” in checks[i]

    Return to main program
}

Function User_ID_is_Valid(userid)
{
    Check to see if userid is one that can be found on this server
    If userid is found on server then
        Return true
    Else
        Return false
}
Function Store_Log(dir, userid)
{
    Create ofstream with "dir" value and a filename
    Get the system time
    Write the system time and the userid requested to the "dir" and filename path
    along with a carriage return
    Return to main program
}

Function Report_Generation()
{
    Set variable "user" = getValue(0) // store userid to user

    Set up enumerated type for the Time Period with Today = 0, Yesterday = 1, Last_Week = 2, Last_Month = 3

    // check to see which Time Period is selected
    Declare variable "search_time"

    If checks[0] is true // Today
        search_time = Today

    Else if checks[1] is true // Yesterday
        search_time = Yesterday

    Else if checks[2] is true // Last Week
        search_time = Last_Week

    Else if checks[3] is true // Last Month
        search_time = Last_Month

    Call Gather_Data(search_time, user)
    Return to main program
}

Function Gather_Data(search_time, user)
{
    // Variables and data structures will be used during execution
    Struct IP which contains a counter and a string
    Struct Files_Referred which contains a counter and a string

    If checks[4] is true // Total Hits is checked
        Initialize variable total_hits to 0
If checks[5] is true // Total Bytes Transferred is checked
    Initialize variable total_bytes to 0

If checks[6] is true // Top Domains Visiting Site
    Create a pointer to an array of pointers to structs
    // This struct will be used for storing the IP addresses and the count for each
telling
    // the number of times they accessed this user's files

If checks[7] is true // Files referenced is checked
    Create a pointer to an array of pointers to structs
    // This struct will be used for storing the file paths accessed and the count for
each
telling the number of times that they were accessed

    // Possible ways to sort files
    Initialize boolean variable quantity to false // Used as a flag for later output
    Initialize boolean variable filename to false // Used as a flag for later output

    If checks[8] is true // Sorted by quantity is true
        quantity = true
    Else if checks[9] is true // Sorted by filename is true
        filename = true

    Initialize boolean variable daily_hits to false // Flags used for later output
    Initialize boolean variable daily_hits_graph to false
    Initialize boolean variable daily_bytes to false
    Initialize boolean variable daily_bytes_graph to false

If checks[10] is true and search_time = Today // Daily hits is checked
    Create array daily_hits_break[1] with only value initialized to 0
Else if checks[10] is true and search_time = Yesterday // Daily hits is checked
    Create array daily_hits_break[1] with only value initialized to 0
Else if checks[10] is true and search_time = Last_Week // Daily hits is checked
    Create array daily_hits_break[7] with all values initialized to 0
Else if checks[10] is true and search_time = Last_Month // Daily hits is checked
    Create array daily_hits_break[31] with all values initialized to 0
If checks[11] is true // Daily hits graph is checked
    daily_hits_graph = true
// Daily hits is not checked yet daily hits graph is checked
If checks[10] is false and checks[11] is true and search_time = Today
    Create array daily_hits_break[1] with only value initialized to 0
Else if checks[10] is false and checks[11] is true and search_time = Yesterday
    Create array daily_hits_break[1] with only value initialized to 0
Else if checks[10] is false and checks[11] is true and search_time = Last_Week
    Create array daily_hits_break[7] with all values initialized to 0
Else if checks[10] is false and checks[11] is true and search_time = Last_Month
    Create array daily_hits_break[31] with all values initialized to 0

If checks[12] is true and search_time = Today  // Daily bytes is checked
    Create array daily_byte_break[1] with only value initialized to 0
Else if checks[12] is true and search_time = Yesterday  // Daily bytes is checked
    Create array daily_byte_break[1] with only value initialized to 0
Else if checks[12] is true and search_time = Last_Week  // Daily bytes is checked
    Create array daily_byte_break[7] with all values initialized to 0
Else if checks[12] is true and search_time = Last_Month  // Daily bytes is checked
    Create array daily_byte_break[31] with all values initialized to 0
If checks[13] is true  // Daily bytes graph is checked
daily_bytes_graph = true

// Daily bytes is not checked yet daily bytes graph is checked
If checks[12] is false and checks[13] is true and search_time = Today
    Create array daily_byte_break[1] with only value initialized to 0
Else if checks[12] is false and checks[13] is true and search_time = Yesterday
    Create array daily_byte_break[1] with only value initialized to 0
Else if checks[12] is false and checks[13] is true and search_time = Last_Week
    Create array daily_byte_break[7] with all values initialized to 0
Else if checks[12] is false and checks[13] is true and search_time = Last_Month
    Create array daily_byte_break[31] with all values initialized to 0

Initialize boolean variable hourly_hits to false  // Flags used for later output
Initialize boolean variable hourly_hits_graph to false
Initialize boolean variable hourly_bytes to false
Initialize boolean variable hourly_bytes_graph to false

If checks[14] is true  // Hourly hits is checked
    Create array hourly_hits_break[24] with all values initialized to 0
If checks[15] is true  // Hourly hits graph is checked
    hourly_hits_graph = true
    // Hourly hits is not checked yet hourly hits graph is checked
If checks[14] is false and checks[15] is true
Create array hourly_hits_break[24] with all values initialized to 0

If checks[16] is true    // Hourly bytes is checked
    Create array hourly_byte_break[24] with all values initialized to 0
If checks[17] is true    // Hourly_bytes graph is checked
    hourly_bytes_graph = true
// Hourly bytes is not checked yet hourly bytes graph is checked
If checks[16] is false and checks[17] is true
    Create array hourly_byte_break[24] with all values initialized to 0

Initialize boolean variable failures to false
If checks[18] is true    // List failures is checked
    Create a pointer to an array of pointers to strings
    // This string will be used for storing the paths that people attempted to access
    but
    // failed

// Opening logs to be read

// Case where search_time is Today
If search_time is Today
{
    // Search the log file that is found at /var/log/httpd/access_log
    // which is the current day’s log
    Open access_log
    While not reaching the end of the file
    {
        Take one entry at a time using the “grep” command to find the user
        // Take each entry that is found and perform the operations on it depending

        // upon what options the user has selected

    // Hits Options
    // Total Hits option is represented by checks[4]
    If checks[4] is true then
        If transfer code is not a failure
            Increment total_hits

    // Daily Hits option is represented by checks[10]
    If checks[10] is true or if checks[10] is false and checks[11] is true then
        If transfer code is not a failure
            Increment daily_hits_break[1]
// Hourly Hits option is present by checks[14]
If checks[14] is true or if checks[14] is false and checks[15] is true then
  If transfer code is not a failure
    Parse the log entry and extract the time from the entry, round it
down to the nearest hour and store it to "hour"
    Increment hourly_hits_break[hour]

// Bytes Transferred Options
// Total Bytes Transferred is represented by checks[5]
If checks[5] is true then
  If bytes transferred field is not blank
    Extract bytes transferred and add to total_bytes

// Daily Bytes option is represented by checks[12]
If checks[12] is true or if checks[12] is false and checks[13] is true then
  If bytes transferred field is not blank
    Extract bytes transferred and add to daily_byte_break[1]

// Hourly Bytes option is represented by checks[16]
If checks[16] is true or if checks[16] is false and checks[17] is true then
  If bytes transferred field is not blank
    Parse the log entry and extract the time from the entry, round
down to the nearest hour and store it to "hour"
    Extract bytes transferred and add to hourly_byte_break[hour]

// Top Domains Visited option
// Top Domains Visiting site is represented by checks[6]
If checks[6] is true then
  Scan the current array of pointers to structs and compare strings with
  the current entry's IP address string
  If matching string is found then
    Increment its count
  Else
    Dynamically allocate a new element in the array that points to a
    struct
    Parse log entry and store the first field or IP address to the string
    value of the IP struct
    Initialize the count to 1

// Files referenced options
// Files referenced is represented by checks[7]
If checks[7] is true then
  If transfer code is not a failure
Scan the current array of pointers to structs and compare strings
with the current entry's file path name
If matching string is found then
    Increment its count
Else
    Dynamically allocate a new element in the array that points
to struct
    Store the files referenced string to the string value of the
files_referenced struct
    Initialize the count to 1

// List Failures option
// List Failures is represented by checks[18]
If checks[18] is true then
    If current entry's code represents a failure
        Scan the current array of pointers to strings and compare them
        with the current entry's files referenced string
        If failure string is not found in the array then
            Dynamically allocate a new element in the array that points
to a string
            Store the current failure files referenced string to the string
value in the array

} Close access_log

// Case where search_time is Yesterday
If search_time is Yesterday
{
    Unzip logfile[0] using the "gunzip" command on UNIX  // logfile[0] is
    Open logfile[0] // Yesterday's log
    While not reaching the end of the file
    {
        Take one entry at a time using the "grep" command to find the user
        // Take each entry that is found and perform the operations on it depending
        // upon what options the user has selected

    // Hits Options
    // Total Hits option is represented by checks[4]
    If checks[4] is true then
        If transfer code is not a failure
            Increment total_hits
// Daily Hits option is represented by checks[10]
If checks[10] is true or if checks[10] is false and checks[11] is true then
  If transfer code is not a failure
    Increment daily_hits_break[1]

// Hourly Hits option is prepresent by checks[14]
If checks[14] is true or checks[14] is false and checks[15] is true then
  If transfer code is not a failure
    Parse the log entry and extract the time from the entry, round
down to the nearest hour and store it to "hour"
    Increment hourly_hits_break[hour]

// Bytes Transferred Options
// Total Bytes Transferred is represented by checks[5]
If checks[5] is true then
  If bytes transferred field is not blank
    Extract bytes transferred and add to total_bytes

// Daily Bytes option is represented by checks[12]
If checks[12] is true or if checks[12] is false and checks[13] is true then
  If bytes transferred field is not blank
    Extract bytes transferred and add to daily_byte_break[1]

// Hourly Bytes option is represented by checks[16]
If checks[16] is true or if checks[16] is false and checks[17] is true then
  If bytes transferred field is not blank
    Parse the log entry and extract the time from the entry, round
down to the nearest hour and store it to "hour"
    Extract bytes transferred and add to hourly_byte_break[hour]

// Top Domains Visited option
// Top Domains Visiting site is represented by checks[6]
If checks[6] is true then
  Scan the current array of pointers to structs and compare strings with
  the current entry's IP address string
  If matching string is found then
    Increment its count
  Else
    Dynamically allocate a new element in the array that points to a
    struct
Parse log entry and store the first field or IP address to the string value of the IP struct
Initialize the count to 1

// Files referenced options
// Files referenced is represented by checks[7]
If checks[7] is true then
  If transfer code is not a failure
    Scan the current array of pointers to structs and compare strings with the current entry's file path name
    If matching string is found, increment its count
  Else
    Dynamically allocate a new element in the array that points to struct
    Store the files referenced string to the string value of the files_referenced struct
    Initialize the count to 1

// List Failures option
// List Failures is represented by checks[18]
If checks[18] is true then
  If current entry's code represents a failure
    Scan the current array of pointers to strings and compare them with the current entry's files referenced string
    If failure string is not found in the array then
      Dynamically allocate a new element in the array that points to a string
      Store the current failure files referenced string to the string value in the array

} Close logfile[0]

// Case where search_time is Last_Week
If search_time is Last_Week
{
  For i = 0 to 6
  {
    Open logfile[i]
    // Scan the opened file
    While not reaching the end of the file
    {
      Take one entry at a time using the "grep" command to find the user
    }
// Take each entry that is found and perform the operations on it depending
// upon what options the user has selected

// Hits Options
// Total Hits option is represented by checks[4]
If checks[4] is true then
   If transfer code is not a failure
      Increment total_hits

// Daily Hits option is represented by checks[10]
If checks[10] is true or if checks[10] is false and checks[11] is true
   If transfer code is not a failure
      Increment daily_hits_break[i]

// Hourly Hits option is prepresent by checks[14]
If checks[14] is true or if checks[14] is false and checks[15] is true
   If transfer code is not a failure
      Parse the log entry and extract the time from the entry,
      round down to the nearest hour and store it to “hour”
      Increment hourly_hits_break[hour]

// Bytes Transferred Options
// Total Bytes Transferred is represented by checks[5]
If checks[5] is true then
   If bytes transferred field is not blank
      Extract bytes transferred and add to total_bytes

// Daily Bytes option is represented by checks[12]
If checks[12] if true or if checks[12] is false and checks[13] is true
   If bytes transferred field is not blank
      Extract bytes transferred and add to daily_byte_break[i]

// Hourly Bytes option is represented by checks[16]
If checks[16] is true or if checks[16] is false and checks[17] is true
   If bytes transferred field is not blank
      Parse the log entry and extract the time from the entry,
      round down to the nearest hour and store it to “hour”
      Extract bytes transferred and add to hourly_byte_break[hour]

// Top Domains Visited option
// Top Domains Visiting site is represented by checks[6]
If checks[6] is true then
   Scan the current array of pointers to structs and compare strings
   with the current entry’s IP address string
   If matching string is found then
      Increment its count
   Else
      Dynamically allocate a new element in the array that points
      to a struct
      Parse log entry and store the first field or IP address to the
      string value of the IP struct
      Initialize the count to 1

// Files referenced options
// Files referenced is represented by checks[7]
If checks[7] is true then
   If transfer code is not a failure
      Scan the current array of pointers to structs and compare
      strings with the current entry’s file path name
      If matching string is found then
         Increment its count
   Else
      Dynamically allocate a new element in the array that points
      to struct
      Store the files referenced string to the string value of the
      files_referenced struct
      Initialize the count to 1

// List Failures option
// List Failures is represented by checks[18]
If checks[18] is true then
   If current entry’s code represents a failure
      Scan the current array of pointers to strings and compare
      them with the current entry’s files referenced string
      If failure string is not found in the array then
         Dynamically allocate a new element in the array that
         points to a string
         Store the current failure files referenced string to the
         string value in the array

}
// Case where search_time is Last_Month
If search_time is Last_Month
{
    For i = 0 to 30
    {
        Open logfile[i]
        // Scan the opened file

        While not reaching the end of the file
        {
            Take one entry at a time using the "grep" command to find the user
            // Take each entry that is found and perform the operations on it depending
            // upon what options the user has selected

            // Hits Options
            // Total Hits option is represented by checks[4]
            If checks[4] is true then
                If transfer code is not a failure
                    Increment total_hits

            // Daily Hits option is represented by checks[10]
            If checks[10] is true or if checks[10] is false and checks[11] is true
                If transfer code is not a failure
                    Increment daily_hits_break[i]

            // Hourly Hits option is represented by checks[14]
            If checks[14] is true or if checks[14] is false and checks[15] is true
                If transfer code is not a failure
                    Parse the log entry and extract the time from the entry,
                    round down to the nearest hour and store it to "hour"
                    Increment hourly_hits_break[hour]

            // Bytes Transferred Options
            // Total Bytes Transferred is represented by checks[5]
            If checks[5] is true then
                If bytes transferred field is not blank
                    Extract bytes transferred and add to total_bytes

            // Daily Bytes option is represented by checks[12]
            If checks[12] if true or if checks[12] is false and checks[13] is true
                If bytes transferred field is not blank
                    Extract bytes transferred and add to daily_byte_break[i]
// Hourly Bytes option is represented by checks[16]
If checks[16] is true or if checks[16] is false and checks[17] is true
   If bytes transferred field is not blank
      Parse the log entry and extract the time from the entry,
      round down to the nearest hour and store it to "hour"
      Extract bytes transferred and add to
      hourly_byte_break[hour]

// Top Domains Visited option
// Top Domains Visiting site is represented by checks[6]
If checks[6] is true then
   Scan the current array of pointers to structs and compare strings
      with the current entry
   If matching string is found then
      Increment its count
   Else
      Dynamically allocate a new element in the array that points
      to a struct
      Parse log entry and store the first field or IP address to the
      string value of the IP struct
      Initialize the count to 1

// Files referenced options
// Files referenced is represented by checks[7]
If checks[7] is true then
   If transfer code is not a failure
      Scan the current array of pointers to structs and compare
      strings with the current entry's file path name
      If matching string is found then
         Increment its count
   Else
      Dynamically allocate a new element in the array that points
      to struct
      Store the files referenced string to the string value of the
      files_referred struct
      Initialize the count to 1

// List Failures option
// List Failures is represented by checks[18]
If checks[18] is true then
   If current entry's code represents a failure
      Scan the current array of pointers to strings and compare
      them with the current entry's files referenced string
If failure string is not found in the array then
    Dynamically allocate a new element in the array that
    points to a string
    Store the current failure files referenced string to the
    string value in the array

]);
Close logfile[i]

// Check to see if we need to have a graphic to draw histograms
If daily_hits_graph or daily_bytes_graph is true
    Open INI file
    Parse the file and find the [Appearance] section
        Compare the strings that follow and look for the string “Daily Histogram
        Image=”
        If found then
            Create string variable day_image
            Store the name of the image to be used for daily histograms
        Close INI file

If hourly_hits_graph or hourly_bytes_graph is true
    Open INI file
    Parse the file and find the [Appearance] section
        Compare the strings that follow and look for the string “Hourly Histogram
        Image=”
        If found then
            Create a string variable hour_image
            Store the name of the image to be used for hourly histograms
        Close INI file

// Time to generate the HTML for the final reports page
Create enumerated type Report which is used for 13 types of reports
Define Reports to be TOTAL_HITS = 1
    TOTAL_BYTESTRANSFERRED = 2
    TOP_DOMAINS = 3
    FILES_REFERENCED = 4
    DAILY_HITS = 5
    DAILY_HITS_GRAPH = 6
    DAILY_BYTES = 7
    DAILY_BYTES_GRAPH = 8
    HOURLY_HITS = 9
    HOURLY_HITS_GRAPH = 10
HOURLY_BYTES = 11
HOURLY_BYTES_GRAPH = 12
FAILURES = 13
Create Order[] of 13 elements that are of enumerated type Report
Initialize array elements to 0    // Which is not one of the values in Reports
Initialize variable index to 0

Open INI file
Parse the file and find the [Report Screen] section
While not reaching the end of the file
    Get each line
    If the line is not started with a semicolon
       Output it to the browser
    If a $ sign is not encountered then
       Output the text to the browser
    Else if a $ sign is encountered
       // Compare string that follows the $ sign up to the next space with
       // possible reports

    If string following $ equates to a total hits report and checks[4] is true
       Store TOTAL_HITS to Order[index]
       Increment index

    Else if string following $ equates to a total bytes transferred report and
    checks[5] is true
       Store TOTAL_BYTES_TRANSFERRED to Order[index]
       Increment index

    Else if string following $ equates to a top domains report and checks[6] is
    true
       Store TOP_DOMAINS to Order[index]
       Increment index

    Else if string following $ equates to a files referenced report and checks[7]
    is true
       Store FILES REFERENCED to Order[index]
       Increment index

    Else if string following $ equates to a daily hits report and checks[10] is
    true
       Store DAILY_HITS to Order[index]
       Increment index
Else if string following $ equates to a daily hits graph and daily_hits_graph is true
    Store DAILY_HITS_GRAPH to Order[index]
    Increment index

Else if string following $ equates to a daily bytes report and checks[12] is true
    Store DAILY_BYTES to Order[index]
    Increment index

Else if string following $ equates to a daily hits graph and daily_bytes_graph is true
    Store DAILY_BYTES_GRAPH to Order[index]
    Increment index

Else if string following $ equates to a hourly hits report and checks[14] is true
    Store HOURLY_HITS to Order[index]
    Increment index

Else if string following $ equates to a hourly hits graph and hourly_hits_graph is true
    Store HOURLY_HITS_GRAPH to Order[index]
    Increment index

Else if string following $ equates to a hourly bytes report and checks[16] is true
    Store HOURLY_BYTES to Order[index]
    Increment index

Else if string following $ equates to a hourly bytes graph and hourly_bytes_graph is true
    Store HOURLY_BYTES_GRAPH to Order[index]
    Increment index

Else if string following $ equates to a failures report and checks[18] is true
    Store FAILURES to Order[index]
    Increment index

For i = 0 to 13 -1
{
    If Order[i] is not equal to zero
        If Order[i] is equal to TOTAL_HITS
Call Total_Hits(total_hits)
Else if Order[i] is equal to TOTAL_BYTES_TRANSFERRED
   Call Total_Bytes_Transferred(total_bytes)
Else if Order[i] is equal to TOP_DOMAINS
   Call Top_Domains(reference to array of structs)
Else if Order[i] is equal to FILES_REFERENCED
   Call Files_Referred(reference to array of structs, quantity)
Else if Order[i] is equal to DAILY_HITS
   Call Daily_Hits(reference to daily_hits_break[])
Else if Order[i] is equal to DAILY_HITS_GRAPH
   Call Daily_Hits_Graph(reference to daily_hits_break[],
                          day_image)
Else if Order[i] is equal to DAILY_BYTES
   Call Daily_Bytes(reference to daily_byte_break[])
Else if Order[i] is equal to DAILY_BYTES_GRAPH
   Call Daily_Bytes_Graph(reference to daily_byte_break[],
                          day_image)
Else if Order[i] is equal to HOURLY_HITS
   Call Hourly_Hits(reference to hourly_hits_break[])
Else if Order[i] is equal to HOURLY_HITS_GRAPH
   Call Hourly_Hits_Graph(reference to hourly_hits_break[],
                          hour_image)
Else if Order[i] is equal to HOURLY_BYTES
   Call Hourly_Bytes(reference to hourly_byte_break[])
Else if Order[i] is equal to HOURLY_BYTES_GRAPH
   Call Hourly_Bytes_Graph(reference to hourly_byte_break[],
                          hour_image)
Else if Order[i] is equal to FAILURES
   Call Failures(reference to array of structs)
{
   Return to program
}

Function Total_Hits(total_hits)
{
   Output HTML code for table with one entry with the label "Total Hits" and with
   total_hits value stored in the one entry box

   Return to program
}

Function Total_Bytes_Transferred(total_bytes)
{

Output HTML code for table with one entry with the label “Total Bytes Transferred” and with total_bytes value stored in the one entry box

Return to program

Function Top_Domains(reference to array of structs)
{
    Output HTML label “Top Domains Accessing This UserID”
    Open INI file
    Parse INI file and find the section [Appearance]
    Parse section until string “Number of Top Sites =” is found
    Take ASCII text after and convert from ASCII to integer value using “atoi” function
    Store value to variable temp
    Close INI file
    Scan array of structs comparing their individual counts and output the top “temp” number of domains by outputting the string data element
    Return to program
}

Function Files_Referred(reference to array of structs, quantity)
{
    If quantity is true then
        Output HTML label “Files referenced: Sorted by quantity”
        Output files referenced strings depending upon their counts
    Else
        Output HTML label “Files referenced: Sorted by name”
        Output files reference strings depending upon their alphabetical names using the UNIX sort function
    Return to program
}

Function Daily_Hits(reference to daily_hits_break[])
{
    Create HTML code for table with number of entries depending upon search_time
    Label table with “Breakdown of Daily Hits”
    Fill in values for table by using the array that was passed and output all HTML code
    Return to program
}

Function Daily_Hits_Graph(reference to daily_hits_break[], day_image)
{ 
    Output HTML label “Daily Breakdown of Hits: Graphical Form”
    Find highest hit count for days in time period and scale all other values by it so that they fit in one screen
    Output HTML code to the browser using the count values as the number of vertical pixels with a set width for each bar and using the day_image as the source
    Output the day number labels below the bars so that it appears as one graphic layout
    Return to program
}

Function Daily_Bytes(reference to daily_byte_break[])
{
    Create HTML code for table with number of entries depending upon search_time
    Label table with “Daily Byte Counts”
    Fill in values for table by using the array that was passed and output all HTML code
    Return to program
}

Function Daily_Bytes_Graph(reference to daily_byte_break[], day_image)
{
    Output HTML label “Daily Breakdown of Bytes Transferred: Graphical Form”
    Find highest byte count for days in time period and scale all other values by it so that they fit in one screen
    Output HTML code to the browser using the count values as the number of vertical pixels with a set width for each bar and using the day_image as the source
    Output the day number labels below the bars so that it appears as one graphic layout
    Return to program
}

Function Hourly_Hits(reference to hourly_hits_break[])
{
    Create HTML code for table that contains entries for 24 hours
    Label the table “Breakdown of Hourly Hits”
    Fill in hit counts for the entries in the table and output all HTML code
    Return to program
}

Function Hourly_Hits_Graph(reference to hourly_hits_break[], hour_image)
{
    Output HTML label “Hourly Breakdown of Hits: Graphical Form”