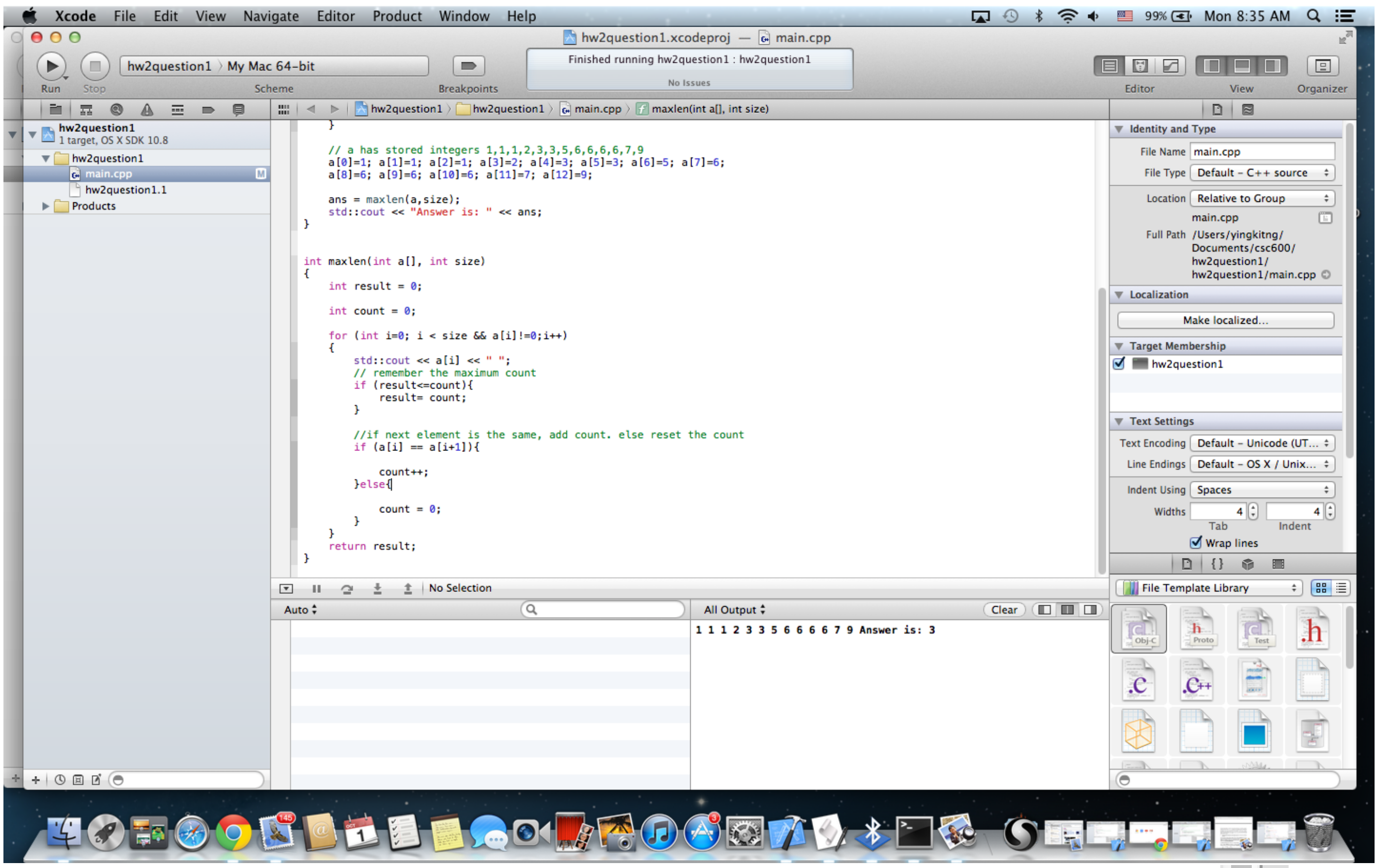
CSC600 11:00-12::00
Homework #2

Ying Kit Ng

Question 1

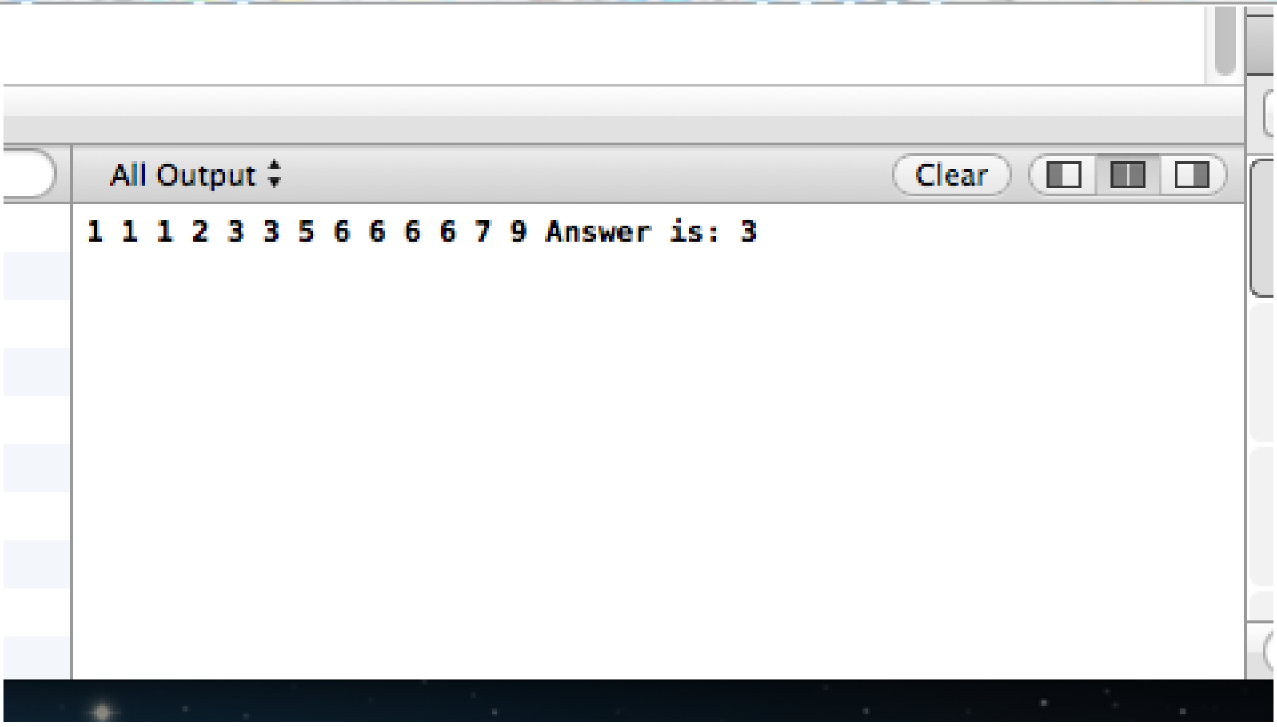
```
//
//  main.cpp
//  CSC675Hw2PlExcercise
//
//  Created by ying kit ng on 9/29/12.
//  Copyright (c) 2012 ying kit ng. All rights reserved.
//

#include <iostream>

/***
 *  the program has single loop in which takes 2 comparsion each time; therefore,
 *  the time complexity is O(n)
 */
int maxlen(int [], int);

int main(){
    int ans;
    int size=20;
    int a[20];

    for (int i=0; i < size;i++)
    {
        a[i]=0;
    }

    // a has stored integers 1,1,1,2,3,3,5,6,6,6,6,7,9
    a[0]=1; a[1]=1; a[2]=1; a[3]=2; a[4]=3; a[5]=3; a[6]=5; a[7]=6;
    a[8]=6; a[9]=6; a[10]=6; a[11]=7; a[12]=9;

    ans = maxlen(a,size);
    std::cout << "Answer is: " << ans;
}


int maxlen(int a[], int size)
{
    int result = 0;

    int count = 0;

    for (int i=0; i < size && a[i]!=0;i++)
    {
```

```
std::cout << a[i] << " ";
// remember the maximum count, increase count again if the larger number is found
if (result<=count){
    result= count;
}

//if next element is the same, add count. else reset the count
if (a[i] == a[i+1]){
```



```
    }

    // a has stored integers 1,1,1,2,3,3,5,6,6,6,6,7,9
    a[0]=1; a[1]=1; a[2]=1; a[3]=2; a[4]=3; a[5]=3; a[6]=5; a[7]=6;
    a[8]=6; a[9]=6; a[10]=6; a[11]=7; a[12]=9;

    ans = maxlen(a,size);
    std::cout << "Answer is: " << ans;
}


int maxlen(int a[], int size)
{
    int result = 0;

    int count = 0;

    for (int i=0; i < size && a[i]!=0;i++)
    {
        std::cout << a[i] << " ";
        // remember the maximum count
        if (result<=count){
            result= count;
        }

        //if next element is the same, add count. else reset the count
        if (a[i] == a[i+1]){

            count++;
        }else{

            count = 0;
        }
    }
    return result;
}
```

```
1 1 1 2 3 3 5 6 6 6 6 7 9 Answer is: 3
```



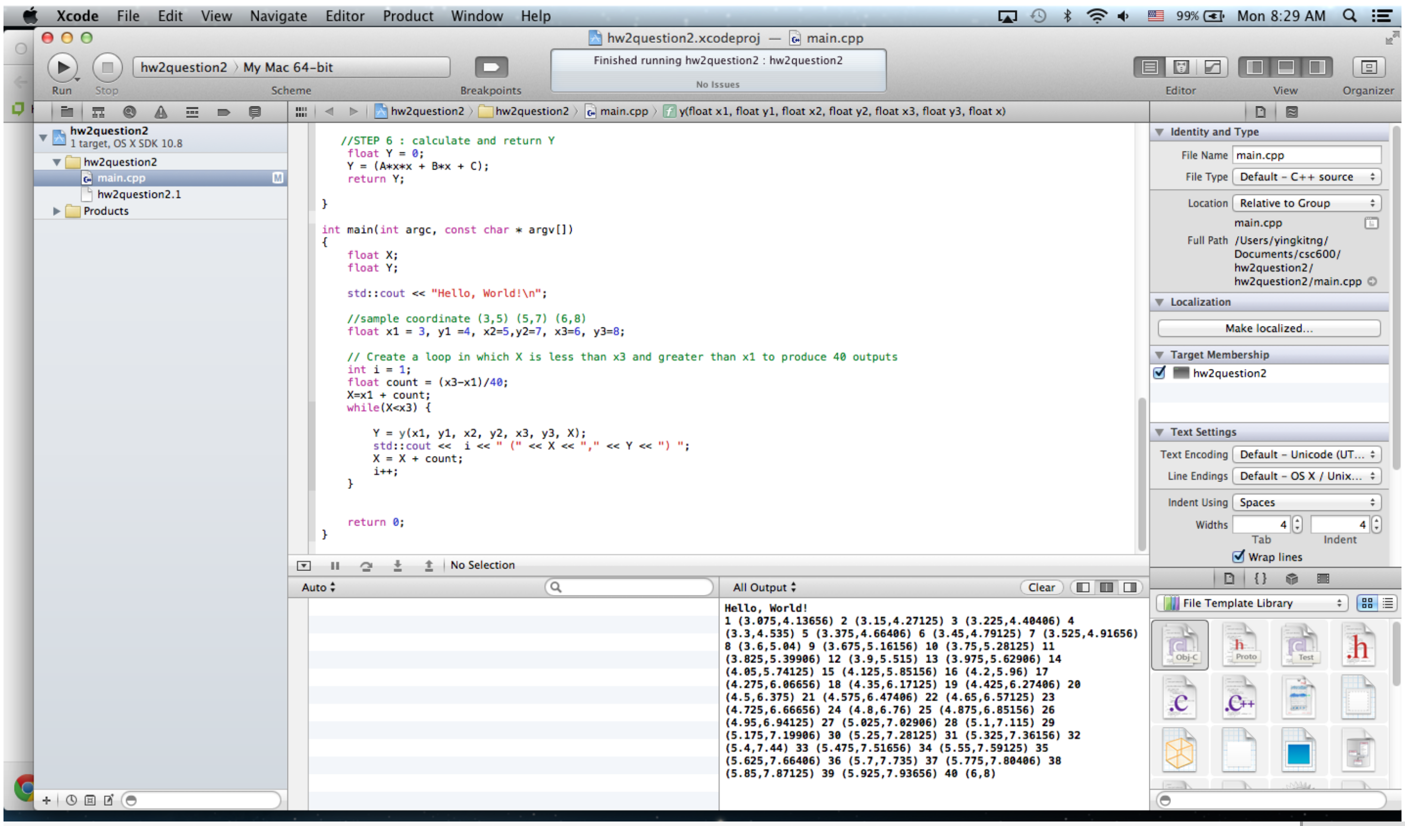
```
1 1 1 2 3 3 5 6 6 6 6 7 9 Answer is: 3
```

Explaination

The program has single loop and the loop takes 2 comparisons each time, so the time complexity is O(n).

Question 2

```
//
//  main.cpp
//  hw2question2
//
//  Created by ying kit ng on 9/29/12.
//  Copyright (c) 2012 ying kit ng. All rights reserved.
//

#include <iostream>

/*
 * Use parabola y = ax2 + bx + c equestion
 * to compute value y.
 *
 * I use the elimiatino method to cancel out the B and C coefficient
 * to get the coefficient A, then I use A to calculate B and C for
 * for the curve.
 */
float y(float x1, float y1, float x2, float y2, float x3, float y3, float x){

    float A = -1, B = 0 , C = 0;
    float CoA[3], CoB[3];

    // STEP 1: I initial coefficient A and B for x1, x2, x3
    // Skip C, because C is elimiated in the following process.
    //  y1 =A(x1)^2+ Bx1 +C
    // -y2=A(x2)^2+ Bx2 +C
    CoA[0]=x1 * x1, CoA[1]=x2 * x2, CoA[2]= x3 * x3;
    CoB[0]=x1, CoB[1]=x2, CoB[2]=x3;

    //std::cout << CoA[0] << " " << CoA[1] << " "  << CoA[2] << " "
     //      << CoB[0] << " "  << CoB[1] << " "  << CoB[2];


    // STEP 2: Use temp variable to store the result of the following
    // y3-y1,y2-y1, A2-A1, B2-B1, A3-A1, B3-B1,
    float tempA = CoA[1] - CoA[0];
    float tempA2 = CoA[2] -CoA[0];
    float tempB = CoB[1] - CoB[0];
    float tempB2 = CoB[2] - CoB[0];

    float tempY = y2 - y1;
    float tempY2 = y3 - y1;

    // STEP3: elimiate B by multipling the opposite coefficient B , then
    float tempAB = tempA2 * tempB - tempA * tempB2;
    float tempYB = tempY2 * tempB - tempY * tempB2;

    A = (tempYB / tempAB);


    //STEP 4: find b with a
    B = (tempY - tempA*A) / tempB;

    //STEP 5: find c with a,b
    C = (y1 - CoA[0]*A - CoB[0]*B);

   //std::cout << " A,B,C = (" << A << ", " << B << "," << C <<") \n";

   //STEP 6 : calculate and return Y
    float Y = 0;
    Y = (A*x*x + B*x + C);
    return Y;

}

int main(int argc, const char * argv[])
{
    float X;
    float Y;

    std::cout << "Hello, World!\n";

    //sample coordinate (3,5) (5,7) (6,8)
    float x1 = 3, y1 =4, x2=5,y2=7, x3=6, y3=8;

    // Create a loop in which X is less than x3 and greater than x1 to produce 40 outputs
    int i = 1;
    float count = (x3-x1)/40;
    X=x1 + count;
    while(X<x3) {

        Y = y(x1, y1, x2, y2, x3, y3, X);
        std::cout <<  i << " (" << X << "," << Y << ") ";
        X = X + count;
        i++;
    }


    return 0;
}
```



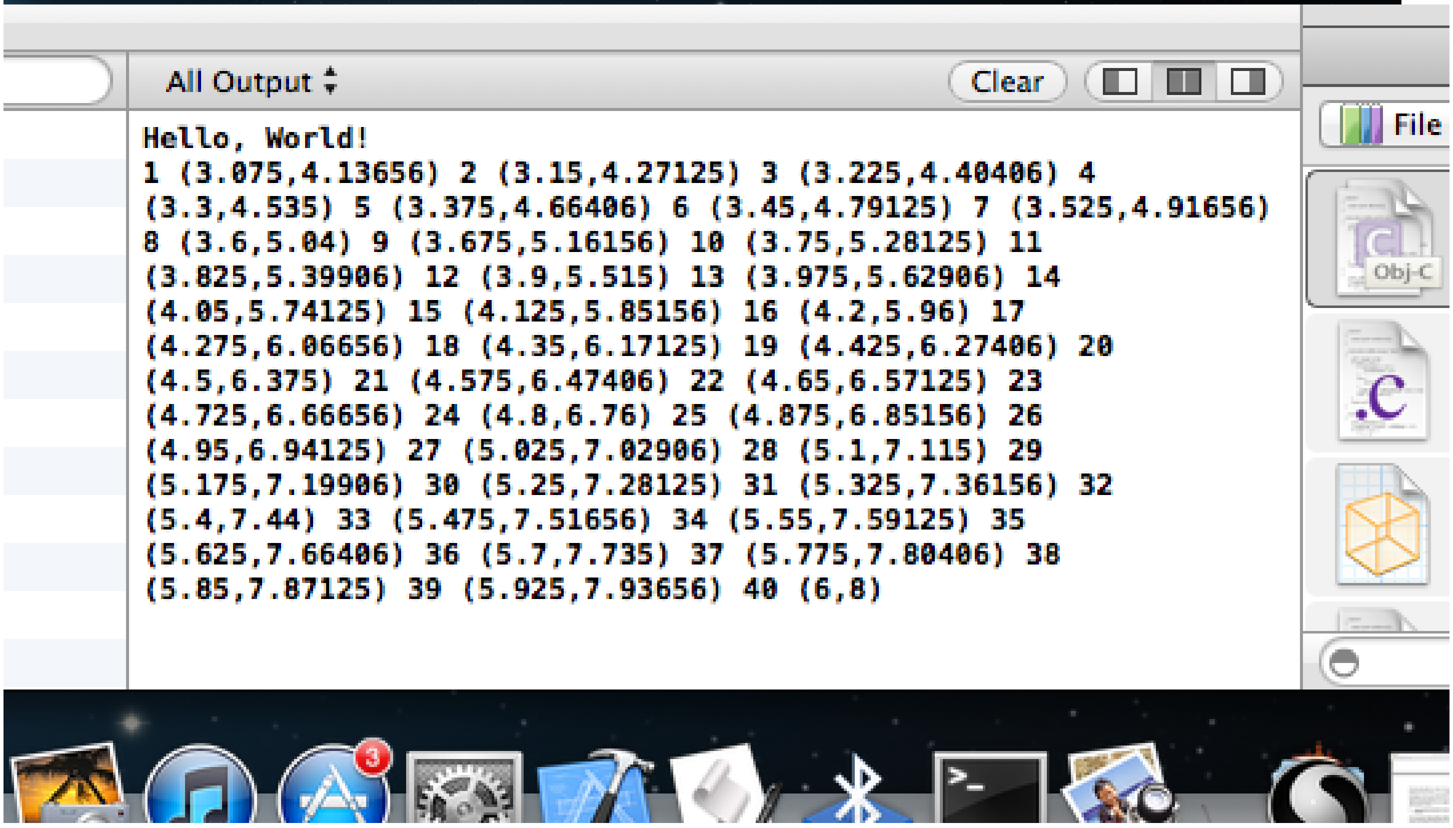
hw2question2.xcodeproj — main.cpp

Finished running hw2question2 : hw2question2

No Issues

```
    //STEP 6 : calculate and return Y
     float Y = 0;
     Y = (A*x*x + B*x + C);
     return Y;

}

int main(int argc, const char * argv[])
{
    float X;
    float Y;

    std::cout << "Hello, World!\n";

    //sample coordinate (3,5) (5,7) (6,8)
    float x1 = 3, y1 =4, x2=5,y2=7, x3=6, y3=8;

    // Create a loop in which X is less than x3 and greater than x1 to produce 40 outputs
    int i = 1;
    float count = (x3-x1)/40;
    X=x1 + count;
    while(X<x3) {

        Y = y(x1, y1, x2, y2, x3, y3, X);
        std::cout <<  i << " (" << X << "," << Y << ") ";
        X = X + count;
        i++;
    }


    return 0;
}
```

```
Hello, World!
1 (3.075,4.13656) 2 (3.15,4.27125) 3 (3.225,4.40406) 4
(3.3,4.535) 5 (3.375,4.66406) 6 (3.45,4.79125) 7 (3.525,4.91656)
8 (3.6,5.04) 9 (3.675,5.16156) 10 (3.75,5.28125) 11
(3.825,5.39906) 12 (3.9,5.515) 13 (3.975,5.62906) 14
(4.05,5.74125) 15 (4.125,5.85156) 16 (4.2,5.96) 17
(4.275,6.06656) 18 (4.35,6.17125) 19 (4.425,6.27406) 20
(4.5,6.375) 21 (4.575,6.47406) 22 (4.65,6.57125) 23
(4.725,6.66656) 24 (4.8,6.76) 25 (4.875,6.85156) 26
(4.95,6.94125) 27 (5.025,7.02906) 28 (5.1,7.115) 29
(5.175,7.19906) 30 (5.25,7.28125) 31 (5.325,7.36156) 32
(5.4,7.44) 33 (5.475,7.51656) 34 (5.55,7.59125) 35
(5.625,7.66406) 36 (5.7,7.735) 37 (5.775,7.80406) 38
(5.85,7.87125) 39 (5.925,7.93656) 40 (6,8)
```



```
Hello, World!
1 (3.075,4.13656) 2 (3.15,4.27125) 3 (3.225,4.40406) 4
(3.3,4.535) 5 (3.375,4.66406) 6 (3.45,4.79125) 7 (3.525,4.91656)
8 (3.6,5.04) 9 (3.675,5.16156) 10 (3.75,5.28125) 11
(3.825,5.39906) 12 (3.9,5.515) 13 (3.975,5.62906) 14
(4.05,5.74125) 15 (4.125,5.85156) 16 (4.2,5.96) 17
(4.275,6.06656) 18 (4.35,6.17125) 19 (4.425,6.27406) 20
(4.5,6.375) 21 (4.575,6.47406) 22 (4.65,6.57125) 23
(4.725,6.66656) 24 (4.8,6.76) 25 (4.875,6.85156) 26
(4.95,6.94125) 27 (5.025,7.02906) 28 (5.1,7.115) 29
(5.175,7.19906) 30 (5.25,7.28125) 31 (5.325,7.36156) 32
(5.4,7.44) 33 (5.475,7.51656) 34 (5.55,7.59125) 35
(5.625,7.66406) 36 (5.7,7.735) 37 (5.775,7.80406) 38
(5.85,7.87125) 39 (5.925,7.93656) 40 (6,8)
```

Explaination

Use the parabola equal y=ax^2+bx to compute value y.

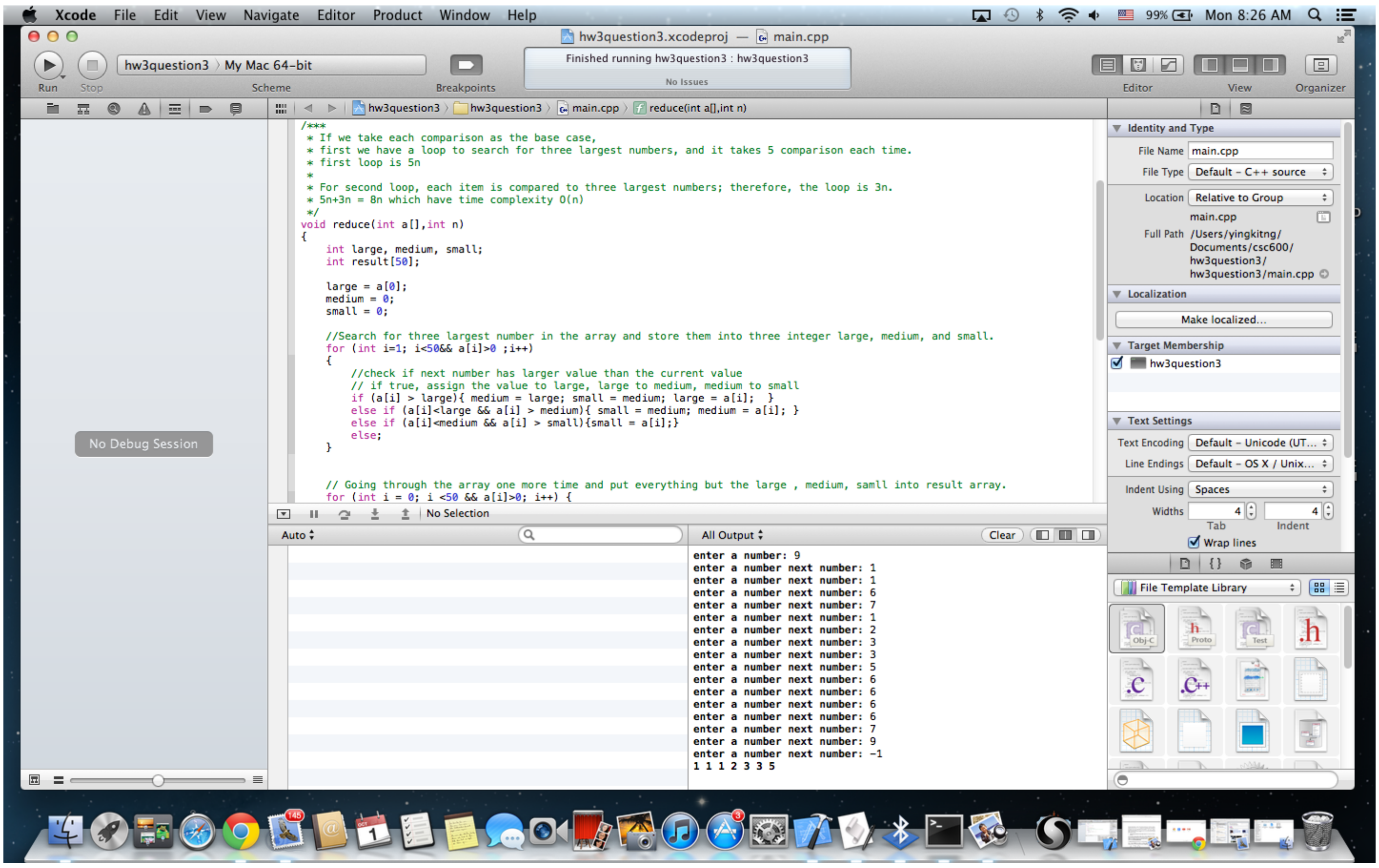
Using the elimiation method to cancel out B and C coeffіcent to get the cofficient A, and use A to retrieve B and C.

Then, the program can calculate y with each input x with A, B, C for n different coordinates.

Question 3

```
//
//  main.cpp
//  hw3question3
//
//  Created by ying kit ng on 9/29/12.
//  Copyright (c) 2012 ying kit ng. All rights reserved.
//

#include <iostream>


/***
 * If we take each comparison as the base case,
 * first we have a loop to search for three largest numbers, and it takes 5 comparison each time.
 * first loop is 5n
 *
 * For second loop, each item is compared to three largest numbers; therefore, the loop is 3n.
 * 5n+3n = 8n which have time complexity O(n)
 */
void reduce(int a[],int n)
{
   int large, medium, small;
   int result[50];

   large = a[0];
   medium = 0;
   small = 0;

   //Search for three largest number in the array and store them into three integer large, medium, and small.
   for (int i=1; i<50&& a[i]>0 ;i++)
   {
      //check if next number has larger value than the current value
      // if true, assign the value to large, large to medium, medium to small
      if (a[i] > large){ medium = large; small = medium; large = a[i];  }
      else if (a[i]<large && a[i] > medium){ small = medium; medium = a[i]; }
      else if (a[i]<medium && a[i] > small){small = a[i];}
      else;
   }


   // Going through the array one more time and put everything but the large , medium, samll into result array.
   for (int i = 0; i <50 && a[i]>0; i++) {
      int j = 0;
```

```
        if (a[i]!=large && a[i]!=medium && a[i]!=small)
        {
            result[j]=a[i];
            std::cout << a[i] << " " ;
            j++;

        }
    }
}

int main(int argc, const char * argv[])
{
    int a[50];

    //initialize the array by inputs
    int i = 0;
    std::cout << "enter a number: ";
    while (std::cin >> a[i] && a[i] > 0 && i <50) {
        std::cout << "enter a number next number: ";
        i++;
    }

    int n = 0 ;
    //call the reduce and display the result
    reduce(a,n);
    return 0;
}
```

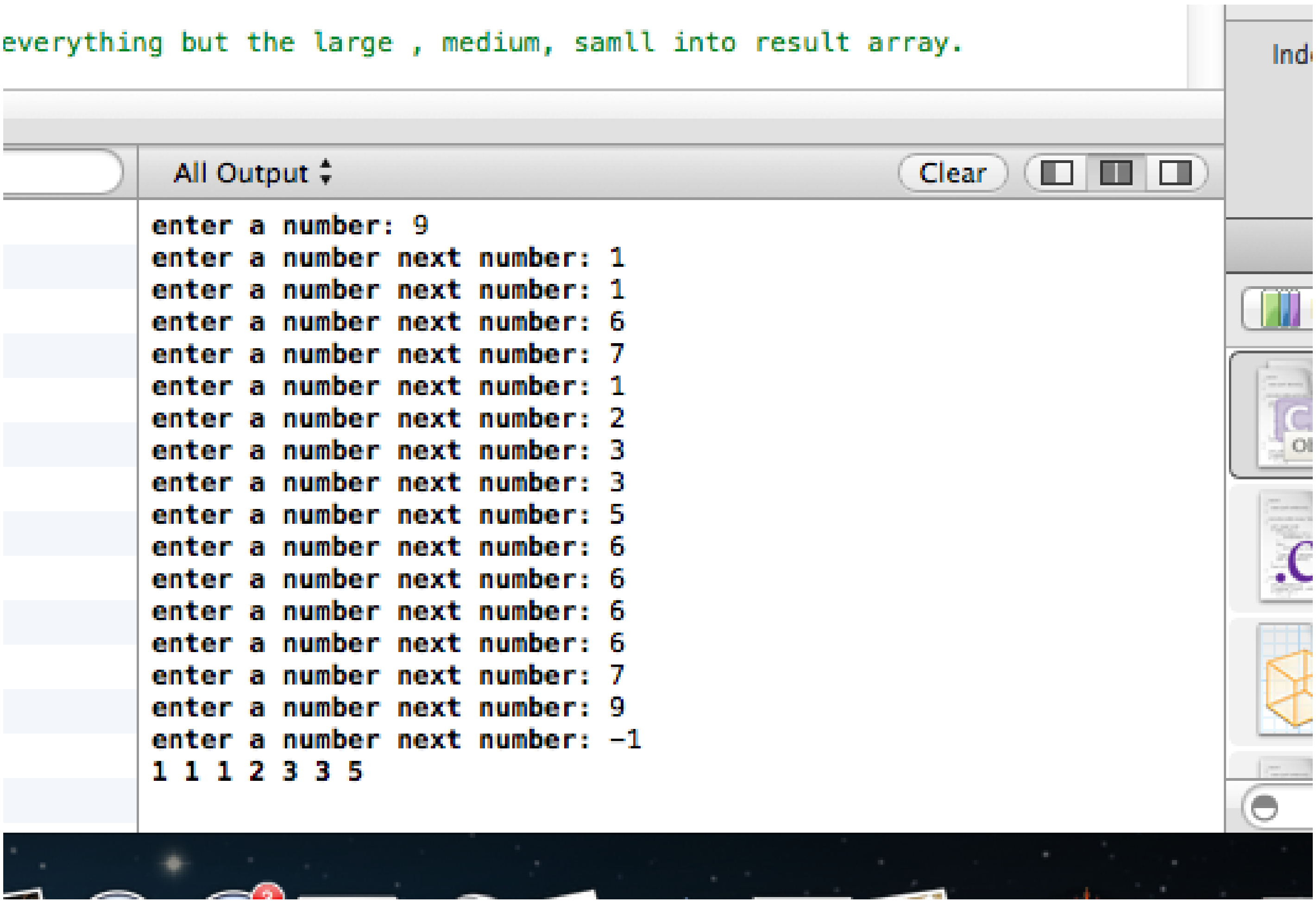


Finished running hw3question3 : hw3question3

hw3question3 > hw3question3 > main.cpp > reduce(int a[],int n)

```cpp
/***
 * If we take each comparison as the base case,
 * first we have a loop to search for three largest numbers, and it takes 5 comparison each time.
 * first loop is 5n
 *
 * For second loop, each item is compared to three largest numbers; therefore, the loop is 3n.
 * 5n+3n = 8n which have time complexity O(n)
 */
void reduce(int a[],int n)
{
    int large, medium, small;
    int result[50];

    large = a[0];
    medium = 0;
    small = 0;

    //Search for three largest number in the array and store them into three integer large, medium, and small.
    for (int i=1; i<50&& a[i]>0 ;i++)
    {
        //check if next number has larger value than the current value
        // if true, assign the value to large, large to medium, medium to small
        if (a[i] > large){ medium = large; small = medium; large = a[i];  }
        else if (a[i]<large && a[i] > medium){ small = medium; medium = a[i]; }
        else if (a[i]<medium && a[i] > small){small = a[i];}
        else;
    }


    // Going through the array one more time and put everything but the large , medium, samll into result array.
    for (int i = 0; i <50 && a[i]>0; i++) {
```

```
enter a number: 9
enter a number next number: 1
enter a number next number: 1
enter a number next number: 6
enter a number next number: 7
enter a number next number: 1
enter a number next number: 2
enter a number next number: 3
enter a number next number: 3
enter a number next number: 5
enter a number next number: 6
enter a number next number: 6
enter a number next number: 6
enter a number next number: 6
enter a number next number: 7
enter a number next number: 9
enter a number next number: -1
1 1 1 2 3 3 5
```



```
everything but the large , medium, samll into result array.
```

```
enter a number: 9
enter a number next number: 1
enter a number next number: 1
enter a number next number: 6
enter a number next number: 7
enter a number next number: 1
enter a number next number: 2
enter a number next number: 3
enter a number next number: 3
enter a number next number: 5
enter a number next number: 6
enter a number next number: 6
enter a number next number: 6
enter a number next number: 6
enter a number next number: 7
enter a number next number: 9
enter a number next number: -1
1 1 1 2 3 3 5
```

Explaination

Taking each comparison as the base case. The program will have 5 comparson each time going through first loop and 3 comparison for second loop.

So the worst case time complexity is 8n which is in O(n)

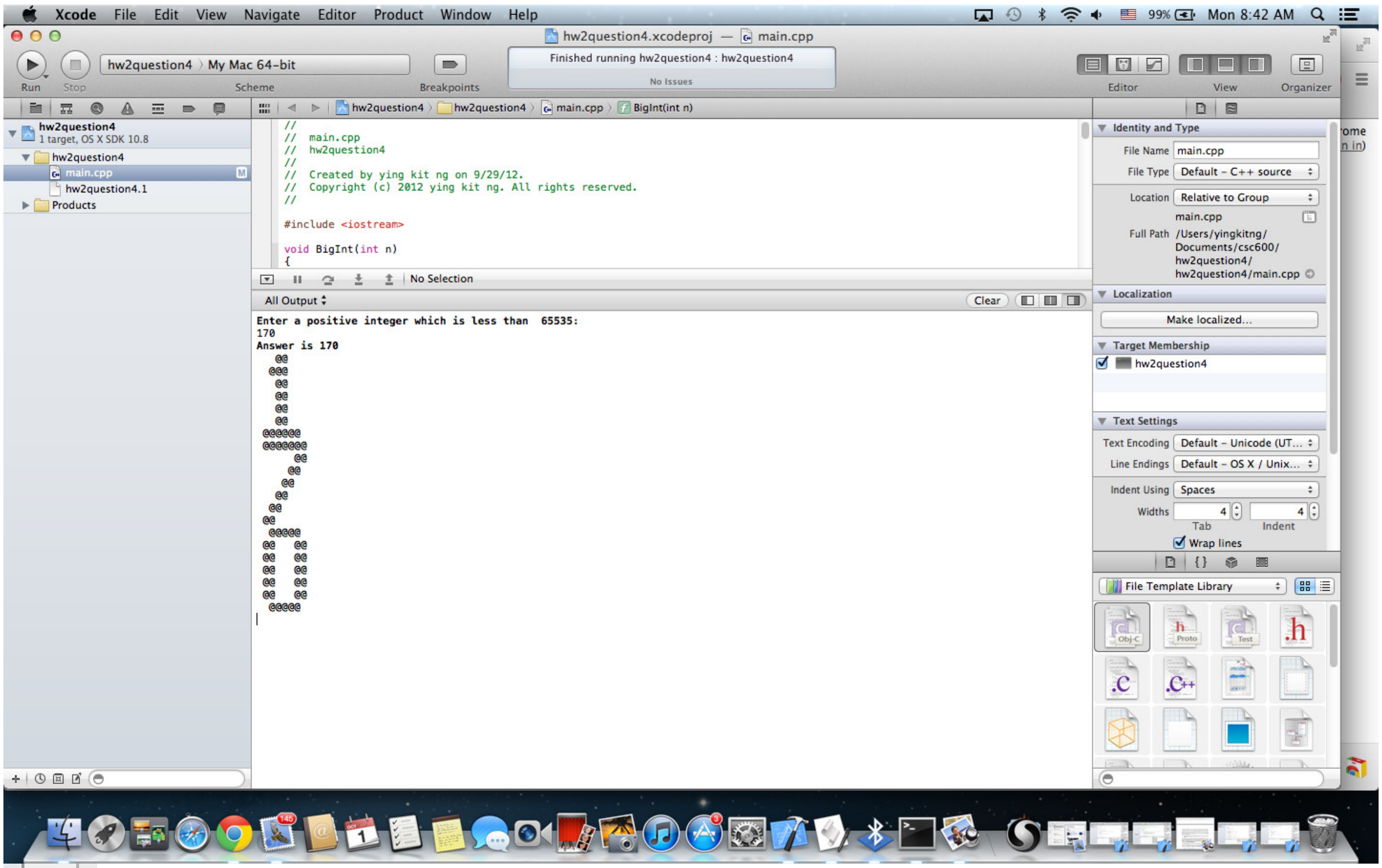
Question 4

```
//
//  main.cpp
//  hw2question4
//
//  Created by ying kit ng on 9/29/12.
//  Copyright (c) 2012 ying kit ng. All rights reserved.
//

#include <iostream>

/***
 * I basely convert the input into a integer array which contains 170
 * then I used the number to display the big character which is constructed
 * by @'s
 */
void BigInt(int n)
{
    int temp = n;
    int bigInt[7];



    std::cout << "Answer is " << temp << " \n";


    for (int i =5;i>=0 ;i--)
    {
        bigInt[i] = temp%10;

        temp = temp /10;
    }

     bool valid = false;
    for (int i = 0; i<6 ; i++)
    {

       // std::cout << bigInt[i] << "\n";

        if (bigInt[i]>0) {
            valid =true;
        }

        if (valid ==true) {
```

```
if (bigInt[i]==0){
    std::cout
    << “  @@@@@   \n”
    << “ @@   @@ \n”
    << “ @@   @@ \n”
    << “ @@   @@ \n”
    << “ @@   @@ \n”
    << “ @@   @@  \n”
    << “  @@@@@  \n”;
}
else if (bigInt[i] == 1){
    std::cout   << “   @@   \n”
                << “  @@@   \n”
                << “   @@   \n”
                << “   @@   \n”
                << “   @@   \n”
                << “   @@   \n”
                << “ @@@@@@ \n”;

} else if (bigInt[i]==2)
{
    std::cout   << “   @@@  \n”
                << “  @  @@ \n”
                << “     @@ \n”
                << “    @@  \n”
                << “   @@   \n”
                << “  @@    \n”
                << “ @@@@@@ \n”;
}else if (bigInt[i]==3)
{
    std::cout   << “   @@@  \n”
                << “  @  @@ \n”
                << “    @@  \n”
                << “   @@   \n”
                << “    @@  \n”
                << “  @  @@ \n”
                << “   @@@  \n”;

}else if (bigInt[i]==4)
{
    std::cout   << “     @  \n”
                << “    @@  \n”
                << “   @@@  \n”
                << “  @ @@  \n”
                << “ @@@@@@ \n”
                << “    @@  \n”
                << “    @@ \n”;
```

```
}else if (bigInt[i]==5)
{
    std::cout
    << “  @@@@@ \n”
    << “  @@    \n”
    << “  @@    \n”
    << “  @@@@@ \n”
    << “     @@ \n”
    << “  @  @@ \n”
    << “  @@@@@ \n”;

}else if (bigInt[i]==6)
{
    std::cout
    << “ @@@@@@ \n”
    << “ @@     \n”
    << “ @@     \n”
    << “ @@@@@@ \n”
    << “ @@  @@ \n”
    << “ @@  @@ \n”
    << “ @@@@@@ \n”;

}else if (bigInt[i]==7)
{
    std::cout
    << “ @@@@@@@ \n”
    << “      @@ \n”
    << “     @@  \n”
    << “    @@   \n”
    << “   @@    \n”
    << “  @@     \n”
    << “ @@      \n”;

}else if (bigInt[i]==8)
{
    std::cout
    << “   @@@  \n”
    << “  @   @ \n”
    << “   @ @  \n”
    << “    @   \n”
    << “   @ @  \n”
    << “  @   @ \n”
    << “   @@@  \n”;

}else if (bigInt[i]==9)
{
    std::cout
    << “  @@@   \n”
    << “ @  @@  \n”
    << “ @  @@  \n”
```

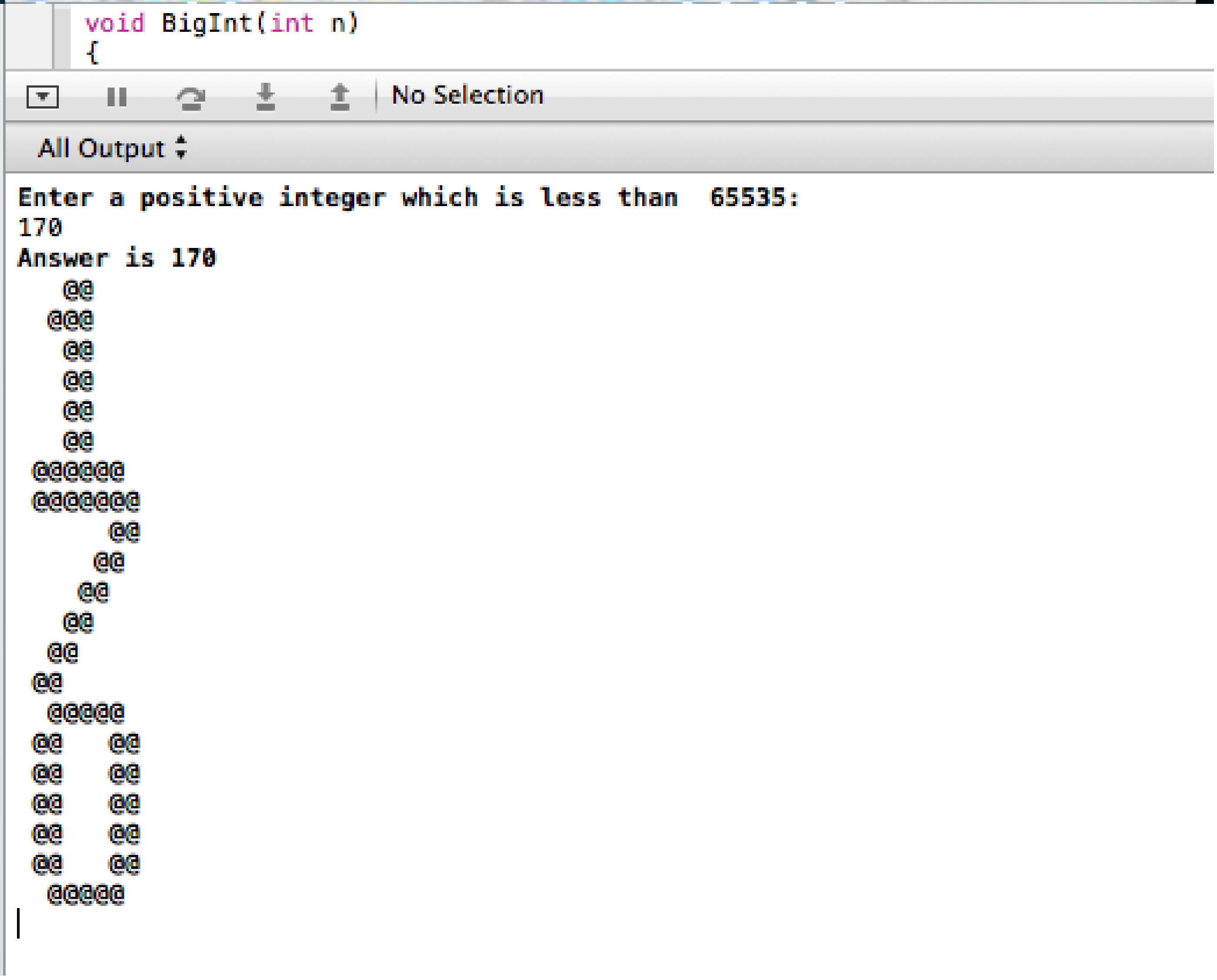
```
            << "  @@@@  \n"
            << "    @@  \n"
            << "    @@  \n"
            << "    @@  \n";

        }else;

        }//end of if(valid)
    }

}

int main(int argc, const char * argv[])
{

    int n;

    // insert code here...
    std::cout << "Enter a positive integer which is less than  65535: \n";

    std::cin >> n;

    BigInt(n);

    return 0;
}
```



File Name: main.cpp
File Type: Default – C++ source
Location: Relative to Group
main.cpp
Full Path: /Users/yingkitng/Documents/csc600/hw2question4/hw2question4/main.cpp
Text Encoding: Default – Unicode (UT…
Line Endings: Default – OS X / Unix…
Indent Using: Spaces
Widths: 4 Tab, 4 Indent
Wrap lines ✓

```
//
//  main.cpp
//  hw2question4
//
//  Created by ying kit ng on 9/29/12.
//  Copyright (c) 2012 ying kit ng. All rights reserved.
//

#include <iostream>

void BigInt(int n)
{
```

```
Enter a positive integer which is less than  65535:
170
Answer is 170
   @@
  @@@
   @@
   @@
   @@
   @@
 @@@@@@
 @@@@@@@
      @@
     @@
    @@
   @@
  @@
 @@
  @@@@@
 @@   @@
 @@   @@
 @@   @@
 @@   @@
 @@   @@
  @@@@@
```



```
void BigInt(int n)
{
```

```
Enter a positive integer which is less than  65535:
170
Answer is 170
   @@
  @@@
   @@
   @@
   @@
   @@
 @@@@@@
 @@@@@@@
      @@
     @@
    @@
   @@
  @@
 @@
  @@@@@
 @@   @@
 @@   @@
 @@   @@
 @@   @@
 @@   @@
  @@@@@
```

Explaination

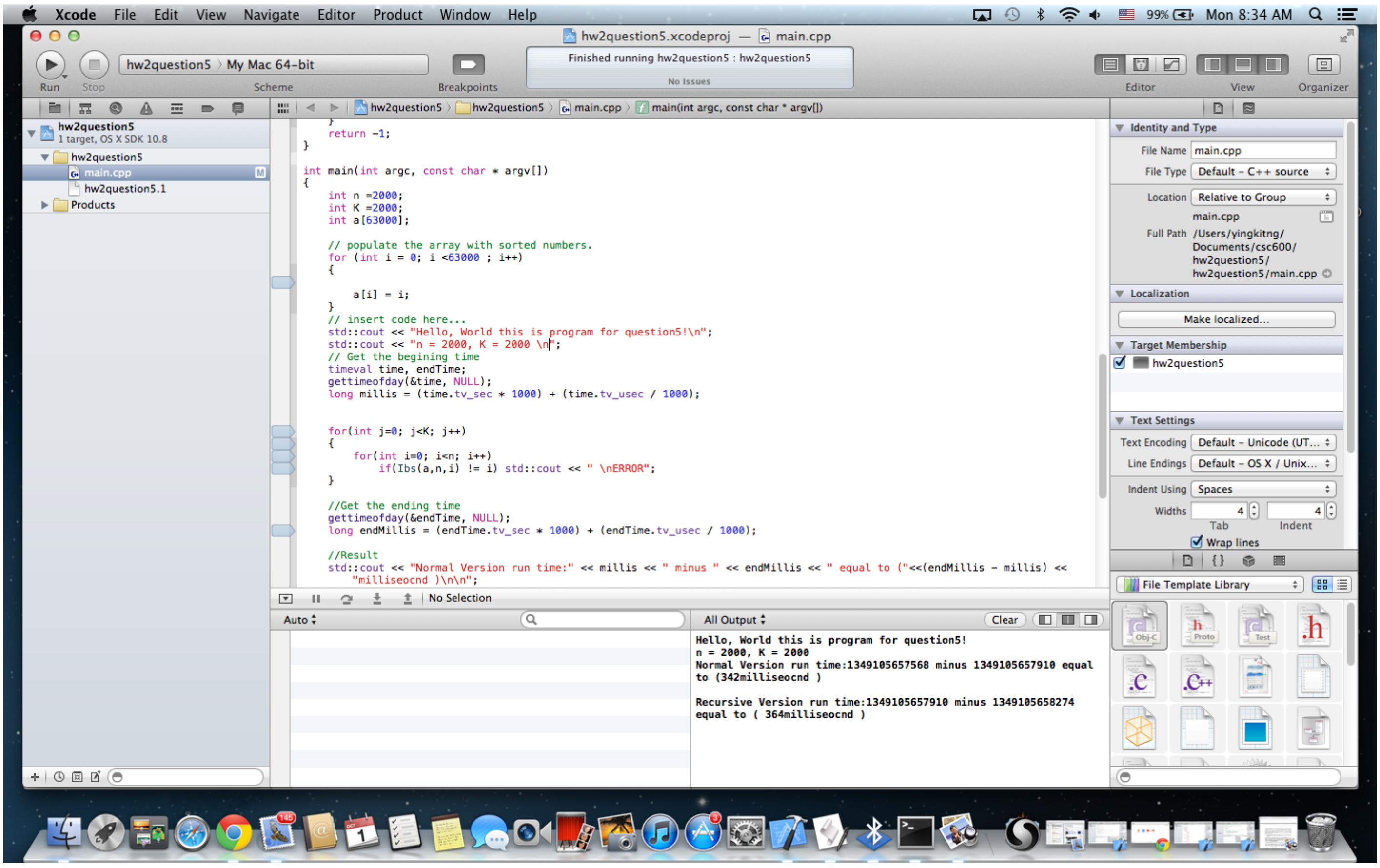
I basely convert the input into a integer array which will contains single digit number.

I display the large character corresponding to each number in the integer array.

Question 5

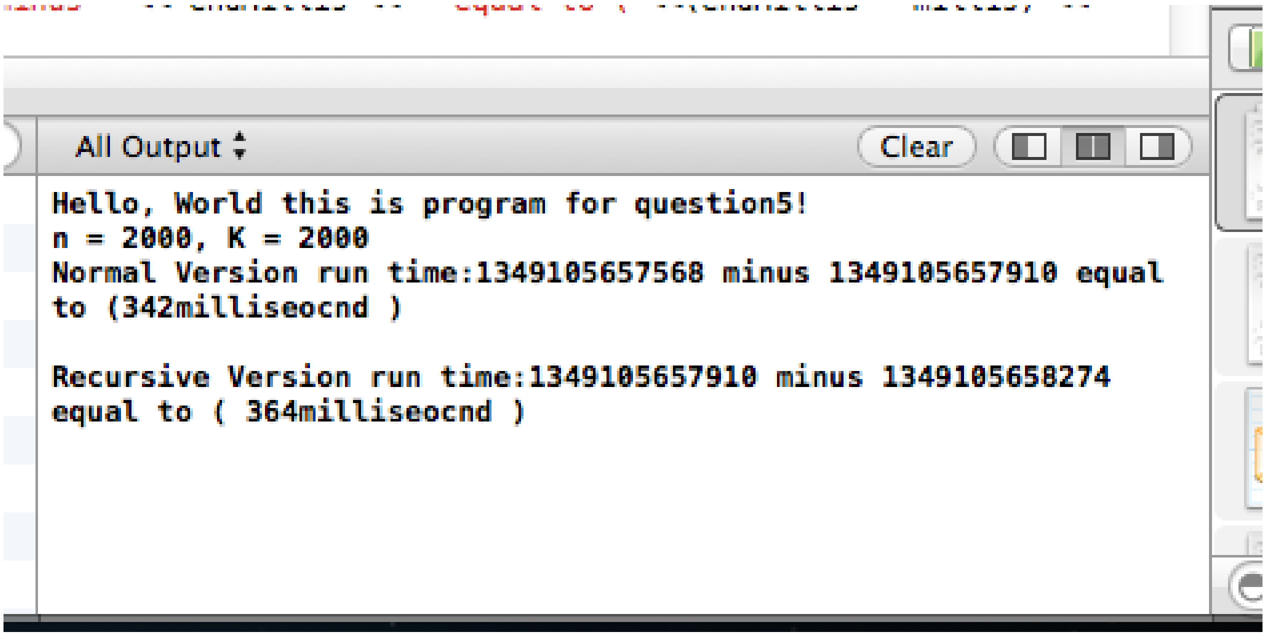
```
//
//  main.cpp
//  hw2question5
//
//  simple program for testing bineary search with a sorted integer array.
//
//  Created by ying kit ng on 9/30/12.
//  Copyright (c) 2012 ying kit ng. All rights reserved.
//

/***
 *  In this program, I create two different verson of bineary serach,
 *  Ibs is linear bineary search, Rbs is Recursive bineary search
 *  In search for large amount of sorted datas, the program shows
 *  the efficient of both arithmetic method
 *  Oddly, the linear bineary search is faster than the Resursive
 *  version, I believe it is because Recursive Bineary Search is a
 *  tail-recursion which no operation are done after recursive call
 *  Iteration version can run faster because no stack needs to be
 *  maintained
 */
#include <iostream>
#include <ctime>
#include <sys/time.h>

// non-recursive bineary search functon
int Ibs(int a[],int n,int x)
{
   int mid = 0, low = 0, high =n;
   int result =0;
   while (low <= high &&  result == 0) {
      // divide the sorted array in half each time
      mid=(low + high)/2;
      if (a[mid] == x)
      {
         result = a[mid];
         return result;
      }else if(x < a[mid]){
         high = mid - 1;
      }else{
         low = mid + 1;
      }
```

```
    }

    // return -1 if the nothing found
    return -1;
}

// recursive bineary search function
int Rbs(int a[],int low, int high,int x)
{
    int mid;
    int result =0;

    if (a[0] == x)
    {
        result = a[0];
        return x;
    }
    else
    {
        mid=(low+high)/2;
        if(x==a[mid])
            return mid;
        else if(x<a[mid])
            return Rbs(a, low, mid-1, x);
        else
            return Rbs(a, mid+1, high, x);
    }
    return -1;
}

int main(int argc, const char * argv[])
{
    int n =2000;
    int K =2000;
    int a[63000];

    // populate the array with sorted numbers.
    for (int i = 0; i <63000 ; i++)
    {

        a[i] = i;
    }
    // insert code here...
    std::cout << "Hello, World this is program for question5!\n";
    std::cout << "n = 2000, K = 2000 \n";
    // Get the begining time
    timeval time, endTime;
    gettimeofday(&time, NULL);
    long millis = (time.tv_sec * 1000) + (time.tv_usec / 1000);
    for(int j=0; j<K; j++)
    {
        for(int i=0; i<n; i++)
            if(Ibs(a,n,i) != i) std::cout << " \nERROR";
    }

    //Get the ending time
    gettimeofday(&endTime, NULL);
    long endMillis = (endTime.tv_sec * 1000) + (endTime.tv_usec / 1000);

    //Result
    std::cout << "Normal Version run time:" << millis << " minus " << endMillis << " equal to ("<<(endMillis - 
millis) << "milliseocnd )\n\n";

    //Start running recursive version
    gettimeofday(&time, NULL);
    millis = (time.tv_sec * 1000) + (time.tv_usec / 1000);

    for(int j=0; j<K; j++)
    {
        for(int i=0; i<n; i++) if(Rbs(a,0, n,i) != i) std::cout << " \nERROR";
    }

    //Get the ending time of recursive version
    gettimeofday(&endTime, NULL);
    endMillis = (endTime.tv_sec * 1000) + (endTime.tv_usec / 1000);

    //Result of second version
    std::cout << "Recursive Version run time:" << millis << " minus " << endMillis << " equal to ( "<<(endMil-
lis - millis) << "milliseocnd )\n";

    return 0;
}
```



```
    return -1;
}

int main(int argc, const char * argv[])
{
    int n =2000;
    int K =2000;
    int a[63000];

    // populate the array with sorted numbers.
    for (int i = 0; i <63000 ; i++)
    {

        a[i] = i;
    }
    // insert code here...
    std::cout << "Hello, World this is program for question5!\n";
    std::cout << "n = 2000, K = 2000 \n";
    // Get the begining time
    timeval time, endTime;
    gettimeofday(&time, NULL);
    long millis = (time.tv_sec * 1000) + (time.tv_usec / 1000);


    for(int j=0; j<K; j++)
    {
        for(int i=0; i<n; i++)
            if(Ibs(a,n,i) != i) std::cout << " \nERROR";
    }

    //Get the ending time
    gettimeofday(&endTime, NULL);
    long endMillis = (endTime.tv_sec * 1000) + (endTime.tv_usec / 1000);

    //Result
    std::cout << "Normal Version run time:" << millis << " minus " << endMillis << " equal to ("<<(endMillis - millis) <<
        "milliseocnd )\n\n";
```

```
Hello, World this is program for question5!
n = 2000, K = 2000
Normal Version run time:1349105657568 minus 1349105657910 equal
to (342milliseocnd )

Recursive Version run time:1349105657910 minus 1349105658274
equal to ( 364milliseocnd )
```



```
Hello, World this is program for question5!
n = 2000, K = 2000
Normal Version run time:1349105657568 minus 1349105657910 equal
to (342milliseocnd )

Recursive Version run time:1349105657910 minus 1349105658274
equal to ( 364milliseocnd )
```

Explaination

In this program , There are two different version of bineary search.

lbs is linear bineary search, Rbs is Recursive bineary search.
In serach for large amount of sorted datas, the program shows the efficient of both method.

The lbs run faster than the Rbs because the lbs doesn't need to maintain a stack while resursive version is a tail resursion and it needs to use the stack.

End of Homework 2 report