```
//Author: Ozan Onder
//C++ Programming Assignments
1)

/*returns the
/length of the longest sequence of identical numbers
/if the array is not empty, the program compares the current
/value of the array with the next element and if they equal, it increments
/sequence. If they are not equal, then the program checks if the maximum lenght is
/less than the sequence and if so, the new new maximum lenght will be the sequence.
/In this case, the vaiable sequence will be initialized to 1 to check the sequence
/relationship of the fursther elements in the array. This procedual solution
/could provide the desired performance O(n) because there is only one loop exists.*/

int maxlen(int a[], int n) //n= number of elements
{
  if(n==0) return 0; //checks whether array is empty

  int sequence = 1;
  int maxlen = 1;//maximum lenght

  for(int i=0; i<=n-1; i++)
  {
    if(a[i]==a[i+1]) //compares the elements
       sequence++;
    else
    {
       if(maxlen<=sequence)//checks if it is the longest
           maxlen=sequence;
       sequence = 1; //initializes sequence for further comparission
    }
  }
  return maxlen; //returns the maximum sequence
}

//main function
int main()
{
    int a[] ={1,1,1,2,3,3,5,6,6,6,6,7,9};
    int n=13;
    cout<<maxlen(a,n)<<endl;

 system("pause");
 return 0;
}

OUTPUT**********

4

3)

/*reduces the array a(1..n) by eliminating from it all values
that are equal to three largest different integers. The procedural programming
```

part of creating this function is that it has three main condition cases. The maximum values are x, y and z, respectively.(x>y>z) First, an array element is retrieved and compared to z. If it is greater than z, then program will check if it is greater than y. If false, it will assigned to z. If true, then the elemenet will be compared to the x. If true it will assigned to x, z will be y and y will be x. If false, then the element will be the new y value and the previous y value will be assigned to z. Therefore, the program follows all the steps condistion by condition. After scanning the loop, three maximim values will be founded and eliminated in the seoncd loop.

The number of the elements n passed as a reference to the function in order to manipulate it. There for, for each removed values, the number n is decremeneted.

```
void reduce(int a[], int &n)
{
  int x=0,y=0,z=0;//max values x>y>z
  for(int i=0; i<n; i++)
  {
     if(a[i]>z)
     {
        if(a[i]>y)
        {
           if(a[i]>x)//checks if the element is the largest
           {
             z=y;//regulates the orders of max values
             y=x;
             x=a[i];
           }
           else if (a[i]!=x)
           {
            z=y;//changes orders of two max values
            y=a[i];
           }
        }
        else if(a[i]!=y)
         z=a[i]; //assigns to the smallest max value
     }
  }//end of for loop
  for(int j=0; j<n; j++)
  {
     if(a[j]==x || a[j]==y ||a[j]==z)
     {
       //if the element is one of the three maximum
       //values, replaces it with zero(eliminates)
       a[j]=0; n--;
     }
     else
       cout<<a[j]<<" ";//displays the valid values
  }
}



//main function
```

```
int main()
{
   int a[]={9,1,1,6,7,1,2,3,3,5,6,6,6,6,7,9};
   int n=16;
   reduce(a,n);


 system("pause");
 return 0;
}
```

OUTPUT**********
1 1 1 2 3 3 5 Press any key to continue . . .

4)

```
  //The following returns the number
//of digits in a given integer
int numDigits(int number)
{
   int digits = 0;
   if (number < 0) digits = 1; // remove this line if '-' counts as a digit
   while (number) {
      number /= 10;
      digits++;
   }
   return digits;
}

//gets the digit of a specified number
//position fromLeft is the leftmost digit of the value
int getDigit(int number, int positionFromLeft)
{
   int posFromRight = 1;
   {
      int n = number;
      while (n /= 10)
         ++posFromRight;
   }
   posFromRight -= positionFromLeft - 1;
   while (--posFromRight)
      number /= 10;
   number %= 10;
   return number > 0 ? number : -number;
}


//Gets the variable number as an argument
//and prints each digit in a 7*7 dimension.

void BigInt(int number)
```

```
{
    int n = 7;  //dimension of each digit
    int r, c; //row and column index of each digit
    int mid = n/2;
    int num_digits = numDigits(number); //# of digits in the argument

    int digits[num_digits];

    //Initialize the array, so that each array element will be
    //each digits of the given nunmber
    for(int i=0; i<num_digits; i++)
      digits[i]=getDigit(number, i+1);

    for(r=1; r<=n; r++, cout << endl)
    {
       for(int i=0; i<num_digits; i++)
       {
          for(c=1; c<=n; c++)
          {
            //The program checks the element of the array
            //and prints a digit depending on the content.
            if(digits[i]==0)
              cout << (c==1 ||c==n || r==1 || r==n ? "@" : " " );
            if(digits[i]==1)
              cout << (c==mid ||c==mid+1 || (c==mid-1 && r==2) ? "@" : " " );
            if(digits[i]==2)
               cout << (r==1 ||r==n || r==mid+1 || (c==n && r<mid+1) || (c==1 && r>mid+1) ? "@" : " " );
            if(digits[i]==3)
               cout << (r==1 ||r==n || r==mid+1 || c==n ? "@" : " " );
            if(digits[i]==4)
               cout << (c==n ||r==n-1 || ((r+c==n-1) &&  r<n) ? "@" : " " );
            if(digits[i]==5)
              cout << (r==1 ||r==n || r==mid+1 || (c==n && r>mid) || (c==1 && r<mid+1) ? "@" : " " );
            if(digits[i]==6)
              cout << (r==1 ||r==n || c==1 || (c==n && r>mid) || r==mid+1 ? "@" : " " );
            if(digits[i]==7)
              cout << (r==1 ||r+c==n+1 || r+c==n+2 ? "@" : " " );
            if(digits[i]==8)
              cout << (r==1 ||r==n || c==1 || c==n || r==mid+1 ? "@" : " " );
            if(digits[i]==9)
              cout << (r==1 ||r==mid+1 || (c==1 && r<mid+1) || c==n ||r==n ? "@" : " " );
          }


       }//end of inner for loop


    }//end of outer for


}//end of the function
```

```
int main()
{

 BigInt(117);
 system("pause");
 return 0;
}
```

OUTPUT**********

```
   @@      @@  @@@@@@@@
 @@@@   @@@@            @@
   @@      @@          @@
   @@      @@         @@
   @@      @@        @@
   @@      @@      @@
   @@      @@    @@
Press any key to continue . . .
```

5-)

```
 #define N 60000 // Array size
#define K 2000 //number of repiton

//time function that is used
//for time measuements of two algorithms

double sec(void)
{
   return double(clock())/CLOCKS_PER_SEC;
}

/*Iterative version of binary search
algorithm that takes 3 arguments
x is the value to be searched.
It finds the medium value of the array and
compares highes and lowest values to it
if it can find, then it returns -1 as return value
*/

int bsearch(int a[], int n, int x)
{ int low, high, mid ;
  low=0 ; high = n-1 ;
  while (low <= high)
  {
     mid = (low + high) / 2 ;
     if (x < a[mid]) high = mid-1 ;
     else if (x > a[mid]) low = mid+1 ;
```

```
    else return mid ; // match
  }
  return -1 ; // no match
}

/*Recursive version of the binary search algorithm
that takes 4 parameters and the value x is the value
to be searched. compares the middle of the array with highest
and lowest values and recalls itself recursively
*/

int bsearch(int a[ ], int low, int high, int x)
{
   int mid = (low + high) / 2;
   if(low>high) return -1; // no match
   if(x<a[mid]) return bsearch(a, low, mid-1, x);//recursive call
   if(x>a[mid]) return bsearch(a, mid+1, high, x);
   return mid; // found!
}

//main function
int main()
{
  int a[N], i,j;
  double t1, t2, t3;//used for the time measuements

  for(i=0; i<N; i++) a[i]=i;//Initialize the array
  t1 = sec();
  for(j=0; j<K; j++)
     for(i=0; i<N; i++) if(bsearch(a, N, i)!=i) cout<< "\nERROR";
  t2 = sec();
  for(j=0; j<K; j++)
  for(i=0; i<N; i++) if(bsearch(a,0,N-1,i)!=i) cout<< "\nERROR";
  t3 = sec();
    cout<< "\nIterative search time = " << t2-t1<< " sec";
    cout<< "\nRecursive search time = " << t3-t2<< " sec\n\n";

 system("pause");
 return 0;
}
```

OUTPUT**********

(AC POWER)
Iterative search time = 25.546 sec
Recursive search time = 26.829 sec

Press any key to continue . . .

(BATTERY ONLY)

Iterative search time = 25.671 sec
Recursive search time = 26.875 sec

Press any key to continue . . .