CS4306 Algorithm Analysis
Fall 2021
Department of Computer Science
Kennesaw State University

Programming Assignment 2
Due Date: Tuesday, September 14, 2021 (by 11:59pm)

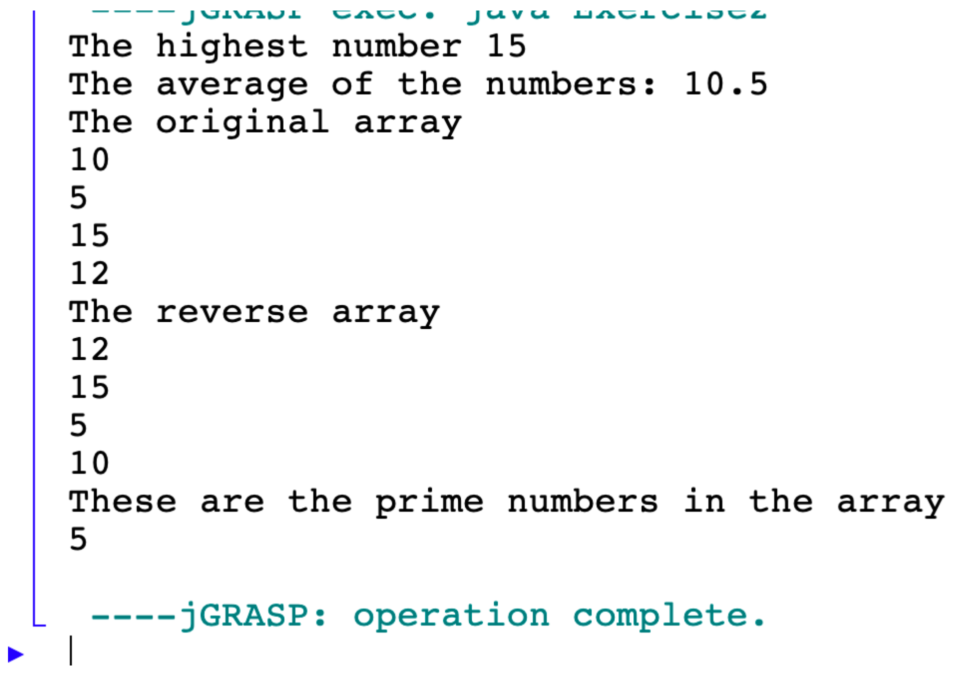
Report

My Anh Huynh

You have been given Java implementation of some sample typical array processing starter code.

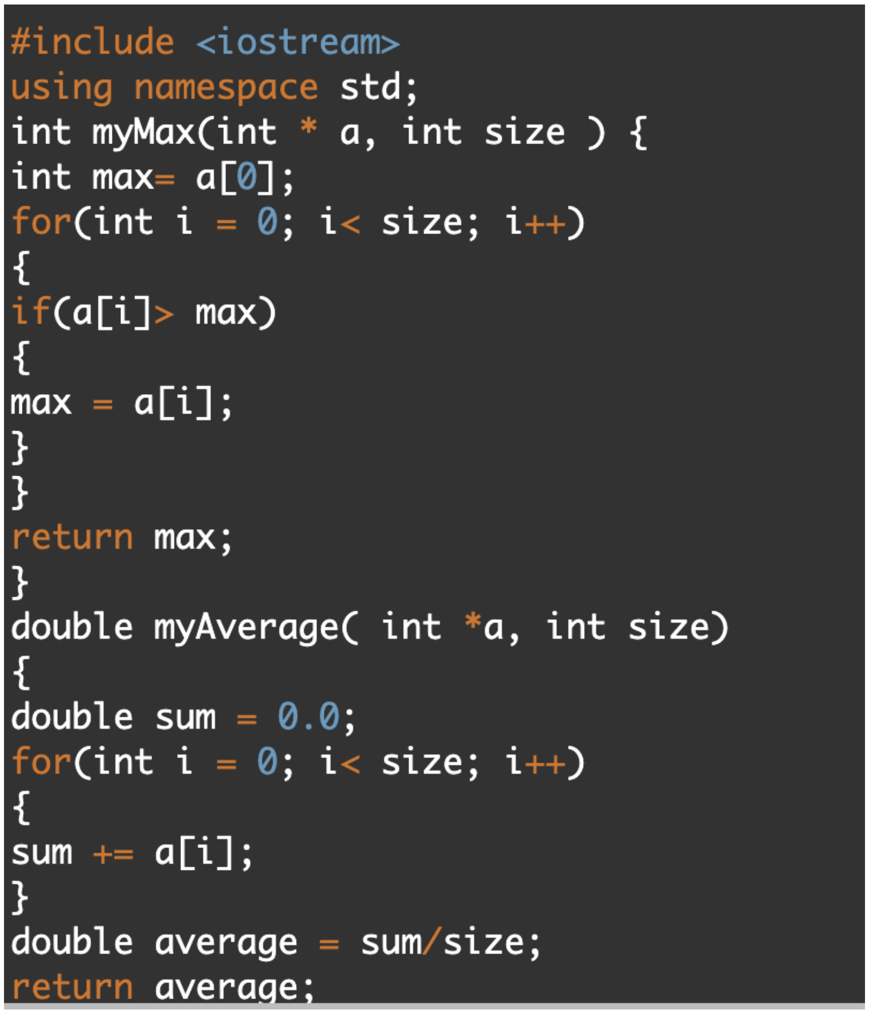
1. First, run the code "as it is", without any modification of the Java code.

2. Attach screen shot of successful run

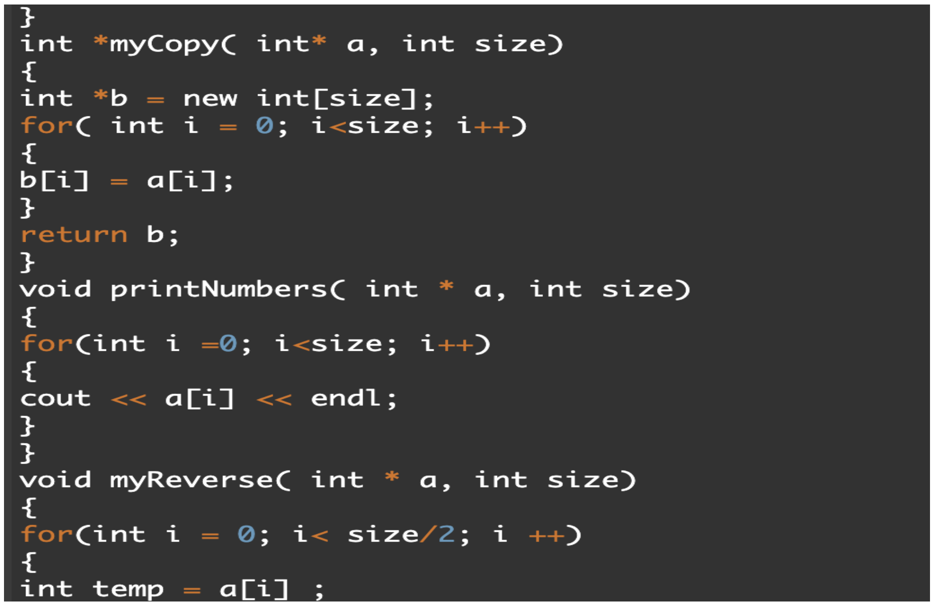


```
The highest number 15
The average of the numbers: 10.5
The original array
10
5
15
12
The reverse array
12
15
5
10
These are the prime numbers in the array
5

 ----jGRASP: operation complete.
```

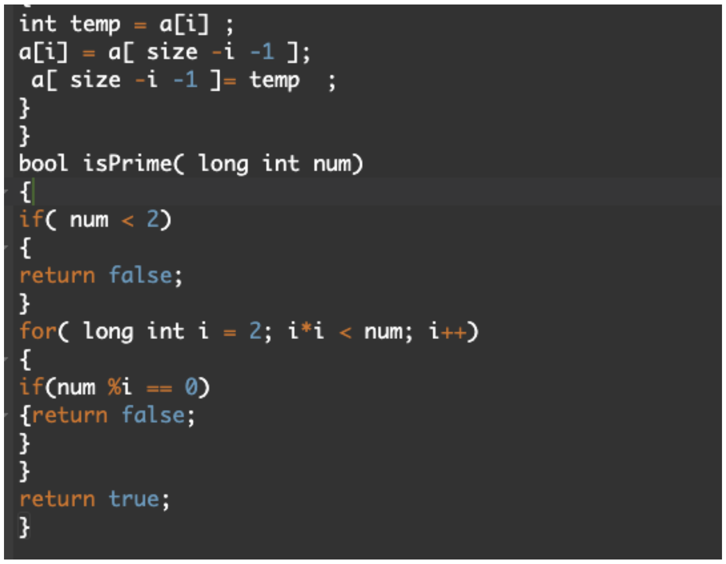
3. Re-write all the functionality using C/C++



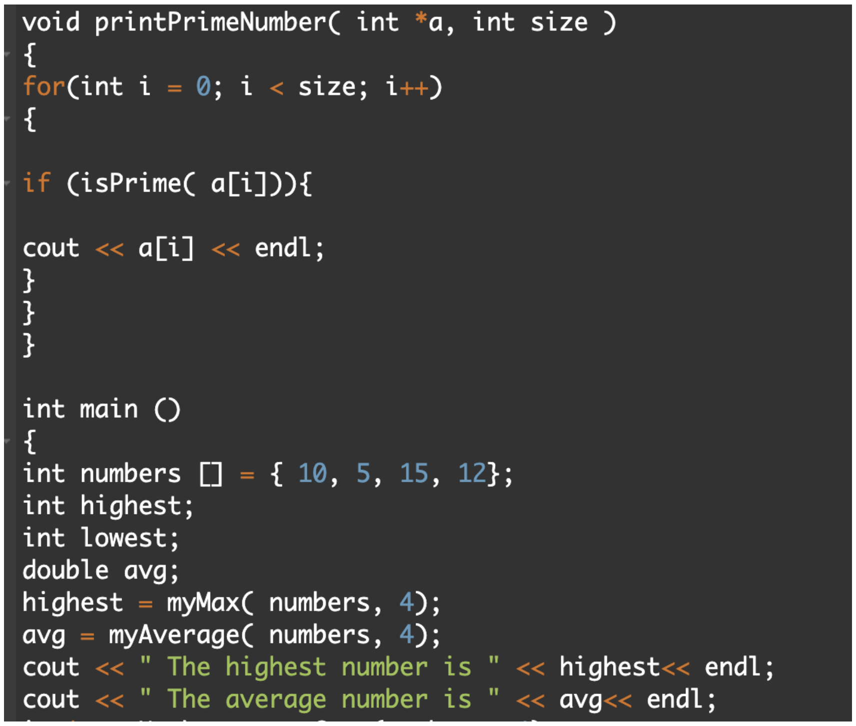
```
#include <iostream>
using namespace std;
int myMax(int * a, int size ) {
int max= a[0];
for(int i = 0; i< size; i++)
{
if(a[i]> max)
{
max = a[i];
}
}
return max;
}
double myAverage( int *a, int size)
{
double sum = 0.0;
for(int i = 0; i< size; i++)
{
sum += a[i];
}
double average = sum/size;
return average;
```



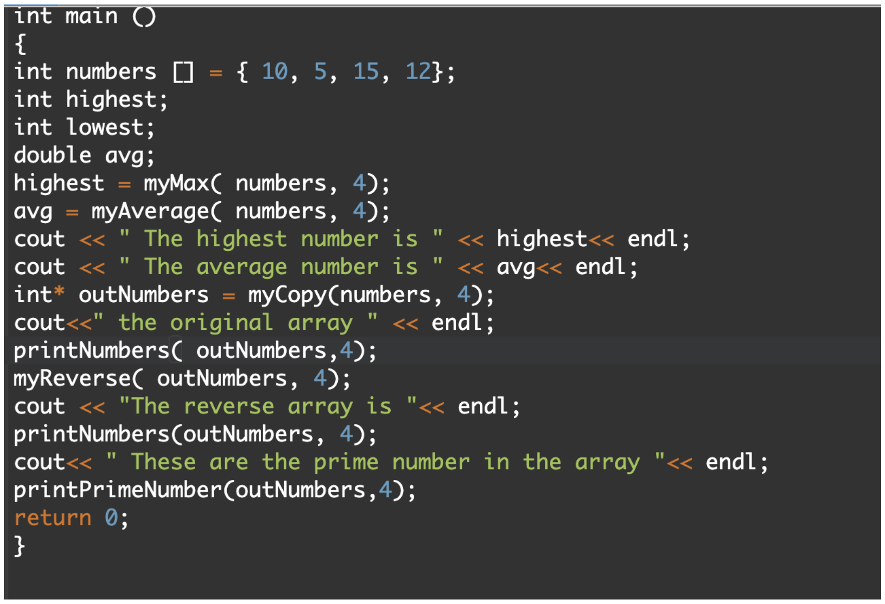
```
}
int *myCopy( int* a, int size)
{
int *b = new int[size];
for( int i = 0; i<size; i++)
{
b[i] = a[i];
}
return b;
}
void printNumbers( int * a, int size)
{
for(int i =0; i<size; i++)
{
cout << a[i] << endl;
}
}
void myReverse( int * a, int size)
{
for(int i = 0; i< size/2; i ++)
{
int temp = a[i] ;
```



```
int temp = a[i] ;
a[i] = a[ size -i -1 ];
 a[ size -i -1 ]= temp  ;
}
}
bool isPrime( long int num)
{
if( num < 2)
{
return false;
}
for( long int i = 2; i*i < num; i++)
{
if(num %i == 0)
{return false;
}
}
return true;
}
```

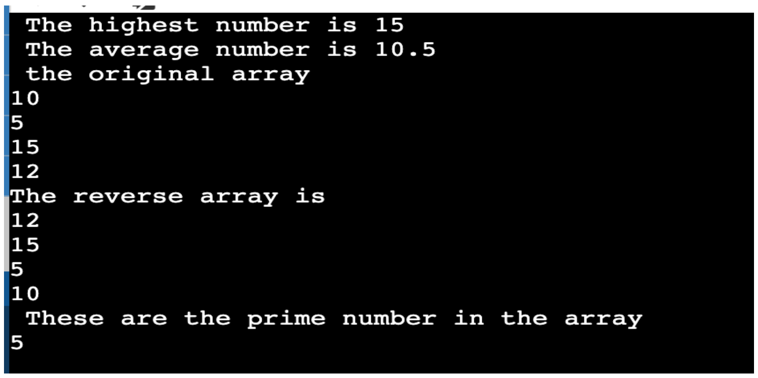


```
void printPrimeNumber( int *a, int size )
{
for(int i = 0; i < size; i++)
{

if (isPrime( a[i])){

cout << a[i] << endl;
}
}
}

int main ()
{
int numbers [] = { 10, 5, 15, 12};
int highest;
int lowest;
double avg;
highest = myMax( numbers, 4);
avg = myAverage( numbers, 4);
cout << " The highest number is " << highest<< endl;
cout << " The average number is " << avg<< endl;
```



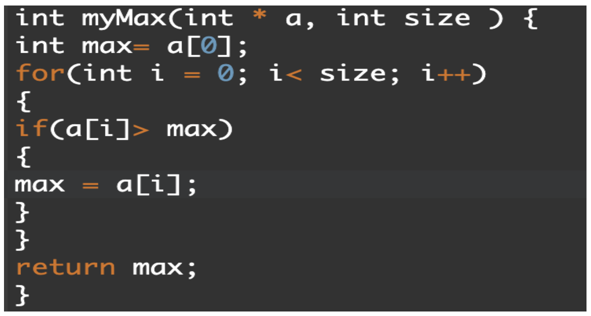
```cpp
int main ()
{
int numbers [] = { 10, 5, 15, 12};
int highest;
int lowest;
double avg;
highest = myMax( numbers, 4);
avg = myAverage( numbers, 4);
cout << " The highest number is " << highest<< endl;
cout << " The average number is " << avg<< endl;
int* outNumbers = myCopy(numbers, 4);
cout<<" the original array " << endl;
printNumbers( outNumbers,4);
myReverse( outNumbers, 4);
cout << "The reverse array is "<< endl;
printNumbers(outNumbers, 4);
cout<< " These are the prime number in the array "<< endl;
printPrimeNumber(outNumbers,4);
return 0;
}
```

4. Again attach scree shot of successful run



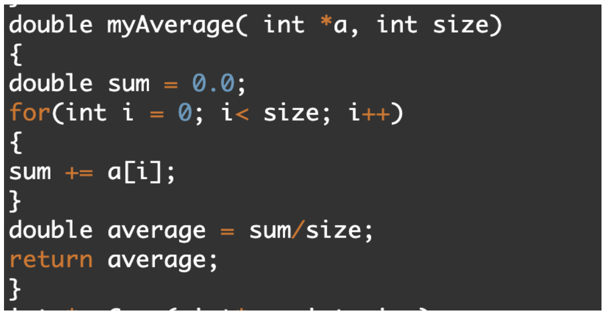
```
 The highest number is 15
 The average number is 10.5
 the original array
10
5
15
12
The reverse array is
12
15
5
10
 These are the prime number in the array
5
```

5. For each operation, analyze running time in terms of Big Oh/Theta notation:



```
int myMax(int * a, int size ) {
int max= a[0];
for(int i = 0; i< size; i++)
{
if(a[i]> max)
{
max = a[i];
}
}
return max;
}
```

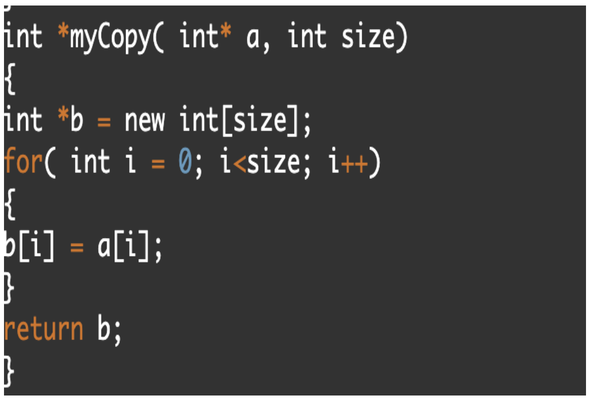
It runs the first time 1 time whenever it is called, and runs the things in the for loop n times. Overall, it runs cn+1 where c is the numbers of thing in the for loop. Hence, this simplifies to O(n) because the cn is the largest term.



```
double myAverage( int *a, int size)
{
double sum = 0.0;
for(int i = 0; i< size; i++)
{
sum += a[i];
}
double average = sum/size;
return average;
}
```

It runs the first time 1 time whenever it is called, and runs the things in the for loop n times. Also, number iteration by inner loop:n

So n*n+1= O(n^2)

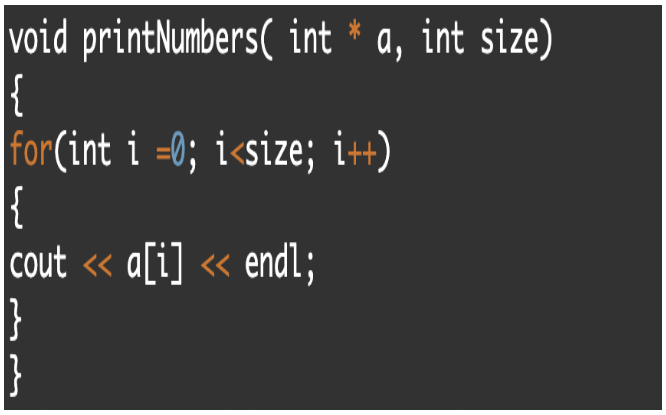


```
int *myCopy( int* a, int size)
{
int *b = new int[size];
for( int i = 0; i<size; i++)
{
b[i] = a[i];
}
return b;
}
```

It runs the first time n time whenever it is called, and runs the things in the for loop n times.

Also, the number of iteration by inner loop is n.

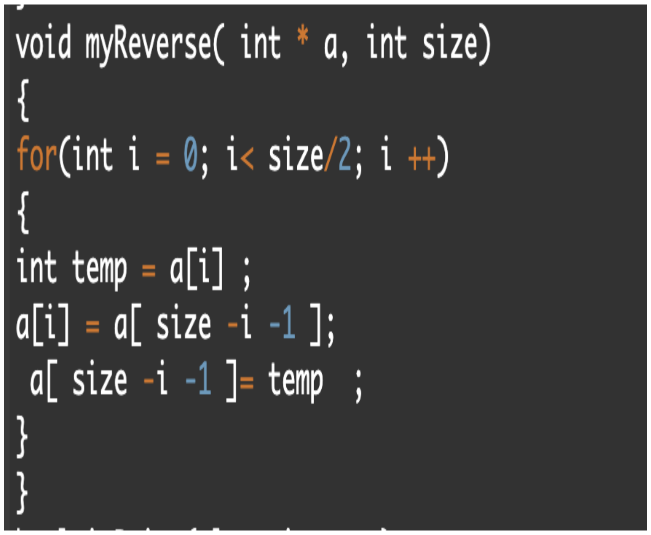
Hence, n+ n*n= O(n^2)



```
void printNumbers( int * a, int size)
{
for(int i =0; i<size; i++)
{
cout << a[i] << endl;
}
}
```

Loop1 is running for i value from 0 to n with increment of 1 so loop1 is running n times.

Loop2 will do the same so O(n*n)= O(n^2)



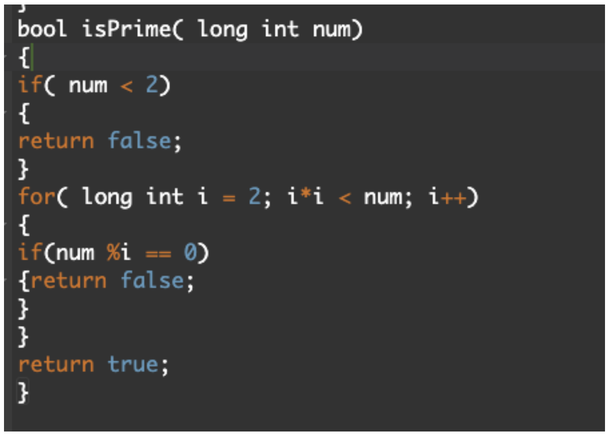
```
void myReverse( int * a, int size)
{
for(int i = 0; i< size/2; i ++)
{
int temp = a[i] ;
a[i] = a[ size -i -1 ];
 a[ size -i -1 ]= temp  ;
}
}
```

Loop1 is running for i value from 0 to n/2 with increment of 1 so loop1 is running n/2 times.

Loop2 is running for I value from 0 to n so running n times.

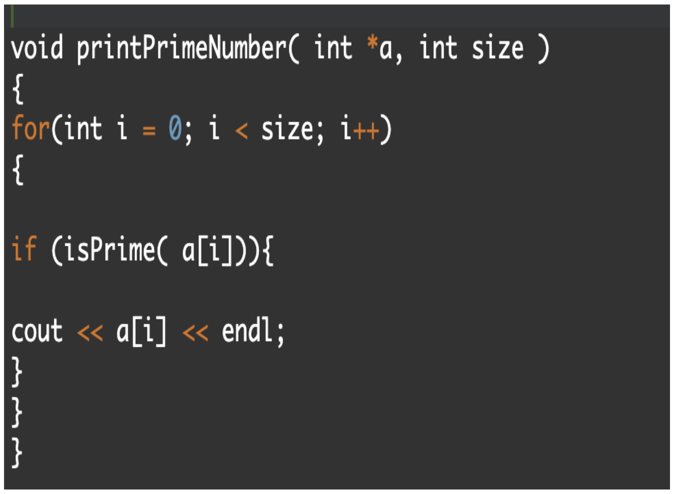
Loop 3 and loop4 is running n time so

n/2 (n*(n+n))= (n/2)*(n*2n)= O(n^3)



```
bool isPrime( long int num)
{
if( num < 2)
{
return false;
}
for( long int i = 2; i*i < num; i++)
{
if(num %i == 0)
{return false;
}
}
return true;
}
```

The first loop is running time 1. The for loop is running (n^(1/2)) times so O(n^(1/2))



```
void printPrimeNumber( int *a, int size )
{
for(int i = 0; i < size; i++)
{

if (isPrime( a[i])){

cout << a[i] << endl;
}
}
}
```

The loop1 running time is n.

The loop2 is running time is n.

Lastly, the loop3 running time is n

Hence, O(n^3)

6. Submit just ONE pdf file that will have everything in it (modified code, screen shot, analysis etc.)