# COMP103 Data Structures and Algorithms

# Tutorial 15.2: Search Algorithms

### A. Fundamentals of Sort Algorithms

1. What is searching and its goal? [2 marks]
2. Why searching is important in programming? [2 marks]
3. What are the criteria used for evaluating searching algorithms? [2 marks]
4. Provide some examples of searching algorithms. [2 marks]

### B. Programming Search Algorithms

1. Complete the program given below which is about binary search (recursive). [2 marks]

   Program:

```
public int indexOf(String value, List<String> data){
return indexOf(value, data, 0, data.size());
}
public int indexOf(String value, List<String> data, int low, int high){
// value in [low .. high)  (if present)
if (low >= high){  return -1;  }              // value not present



if (comp == 0) {
return mid;
}  // item is present
else if (comp < 0)   {
return indexOf(value, data, low, mid);
}  // item in [low .. mid)
else {
return indexOf(value, data, mid+1, high);
}  // item in [mid+1 .. high)
}
```

2. Complete the program given below which is about the binary search (iterative). [4 marks]

Program:

```
private int indexOf(String value, List<String> data){
int low = 0;
int high = data.size();
// item in [low .. high)  (if present)
while (low < high){



if (comp == 0)     // item is at mid
     return mid;
if (comp < 0)      // item in [low .. mid)
                   // item in [low .. high)
else               // item in [mid+1 .. high)
                   // item in [low .. high)
}
return -1;         // item in [low .. high) and low >= high,
}                  // therefore item not present
```

3. The following skeleton of a program is the other form of binary search. [3 marks]

Program:

```
/* Return the index of where the item ought to be, whether present or
not.  (!) */
private int findIndex(String value, List<String> data){
// note: correct position might be at end (index =size)
int low = 0;
int high  =  data.size();      // index in [low .. high]

while (low < high){

    if (value.compareTo(data.get(mid)) > 0)  // index in [mid+1 .. high]
                          // index in [low .. high]  low <= high
    else                  // index in [low .. mid]
                                // index in [low .. high], low<=high
}
return low;               // index in [low .. high] and low = high
                          // therefore index = low
}
```

4. Searching item using bisection algorithm. [7 marks]

Program:

```
bisection( -100, 100, (double x)-> {return (3*x*x*x - 4*x*x + 321);} );

bisection( -100, 100, (x)-> (3*x*x*x - 4*x*x + 321) );


public double bisection(double low, Double high, Function<Double,
Double> function){

double fLow = function.apply(low);

double fHigh = function.apply(high);




return Double.NaN;

}  // same side of axis

while (true) {



if (Math.abs(fMid)<THETA) {return mid;}

else if (Math.signum(fLow) == Math.signum(fMid) ){


}

else {


}

}

}
```