1. bool operator< (specie); // < prototype

bool specie::operator< (specie s){  
    if(population < s.population){  
        return true;  
    } else {  
        return false;  
    }  
}

1. friend ostream & operator<< (ostream &, specie); // << prototype of method that should be public

ostream & operator<<(ostream &os, specie s){

    os << s.name << " has a current population of " << s.population << " and is " << s.status;

    return os;

}

1. bool search(int [], int, int, int); // prototype

bool search(int dataarray[], int x, int low, int high){

int mid;

while(low <= high){

mid = (low + high) / 2;

if(dataarray[mid] == x){

return true;

}

if(dataarray[mid] > x){

high = mid - 1;

} else {

low = mid + 1;

}

}

return false;

}

1. void bnode::infixprint(){

  if(left != NULL){

        left->infixprint();

    }

    cout << data << endl;

    if(right != NULL){

        right->infixprint();

    }

}

1. bool search(int []), int, int); // prototype

bool search(int dataarray[], int x, int len){ // len is the length of dataarray[]

    for(int i = 0; i < len; i++){

        if(dataarray[i] == x){

            return true;

        }

    }

    return false;

}

1. T