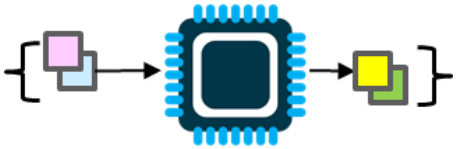
Unit 3d – Array and Loop Tasks

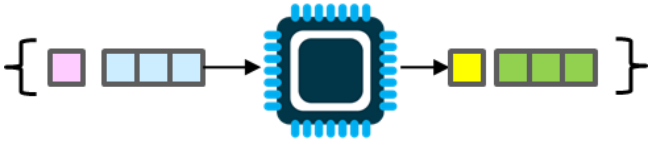
Mark Redekopp

Unit 3

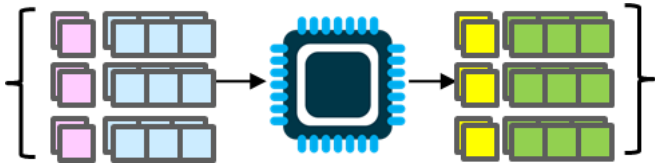
- **Unit 1**: Scalar processing
  - aka IPO=Input-Process-Output Programs



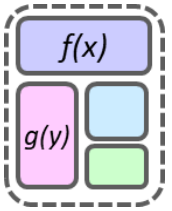
- **Unit 2**: Linear (1D) Processing



- **Unit 3**: Multidimensional Processing



- **Unit 4**: Divide & Conquer (Functional Decomposition)



Algorithmic Thinking

- Informal definition of **algorithm:**
  - **A precise way to accomplish a task or solve a problem**
- The skill we REALLY want to help you build is **algorithmic thinking** (i.e. computational problem solving)

- *We will just try to work as many example as possible but you need to be mentally engaged and trying to solve these tasks before and while we go through them together.*

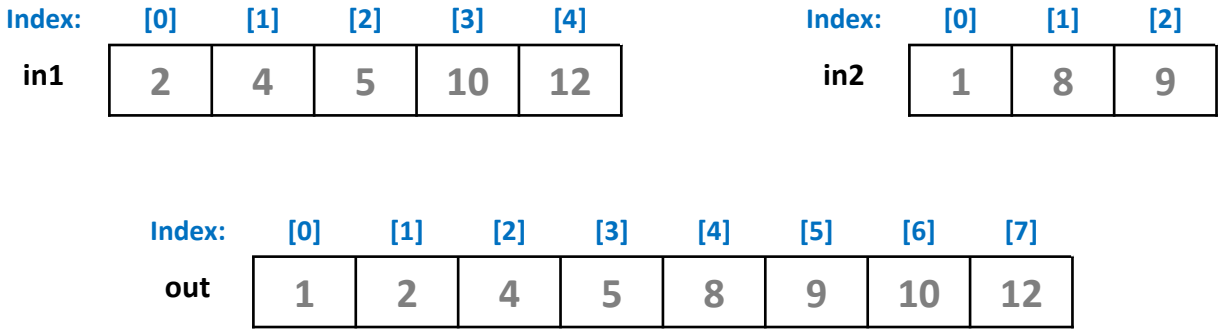
Implementation

- Algorithm Discovery:
  - Solve the problem yourself for several examples
  - Reflect on what your thought process was
  - Given a computer can only do 1 operation (on two values) at a time, what **variables** do you need to remember past results and what **loops** are necessary to perform that 1 operation many times
- Implementation / Programming mechanics
  - Can we achieve our task in one pass (loop), sequential loops, or need nested loops
  - Can we stop early? And how do we implement that (break statements, etc.)

Task 8

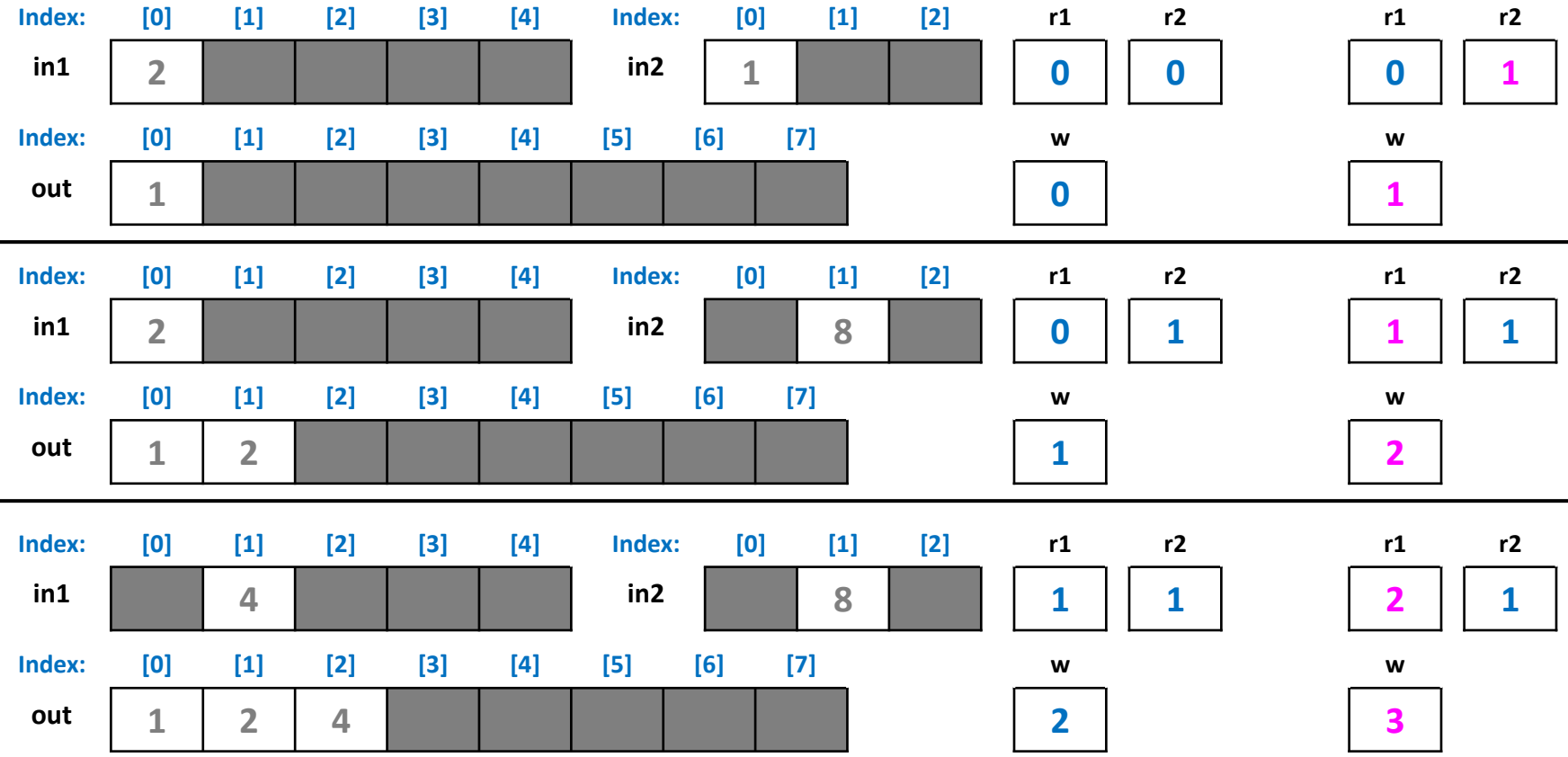
- Given two **sorted** arrays of size n1 and n2 respectively, merge them into a single sorted array of size n1+n2
- Questions:
  - Do we need nested loops or sequential loops?
  - What are the options for whom should in the first output location?

Merge the two sorted arrays into a single sorted array



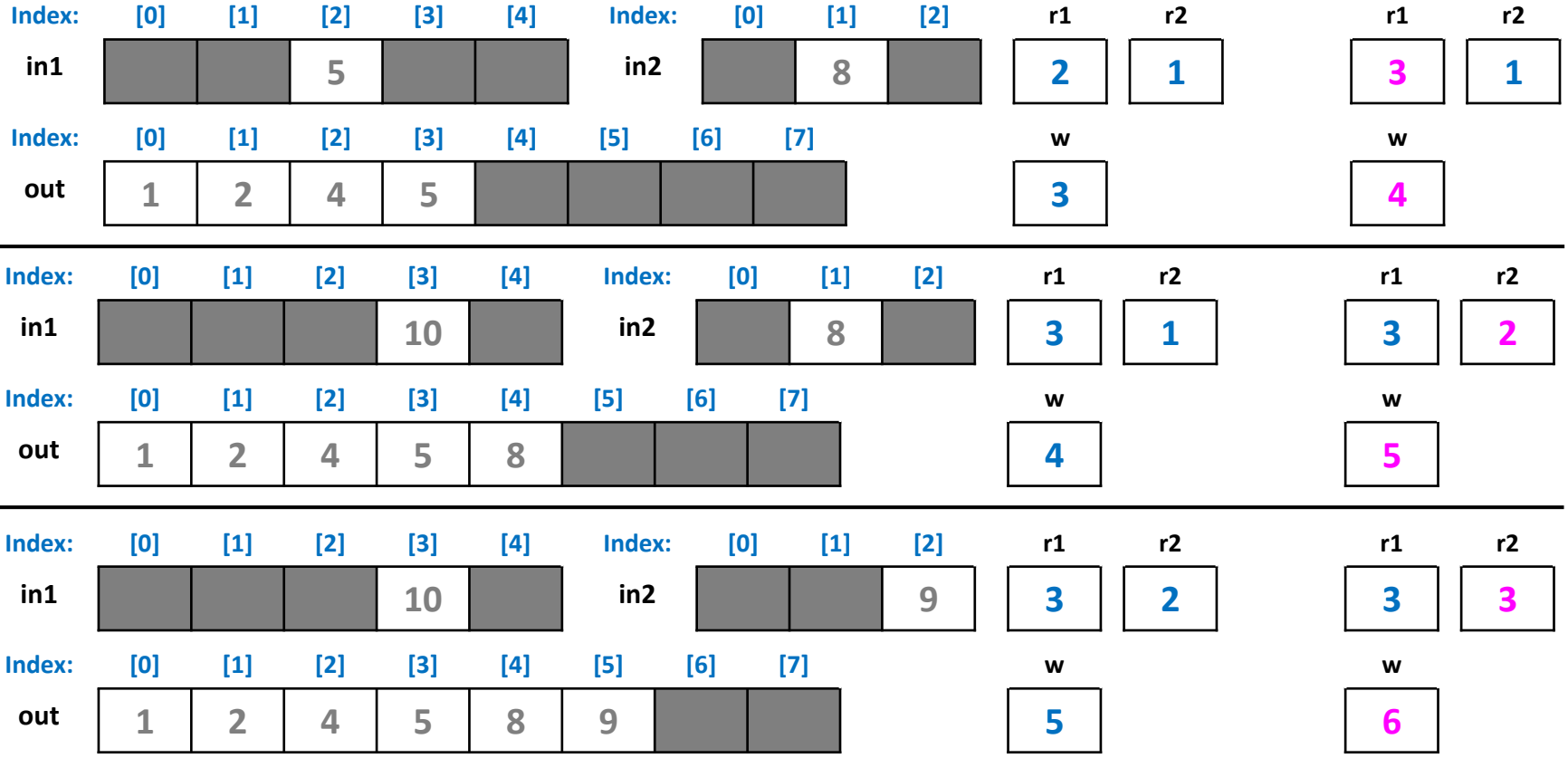
Task 8

If the user enters **3**, find **3** and return its index or -1:



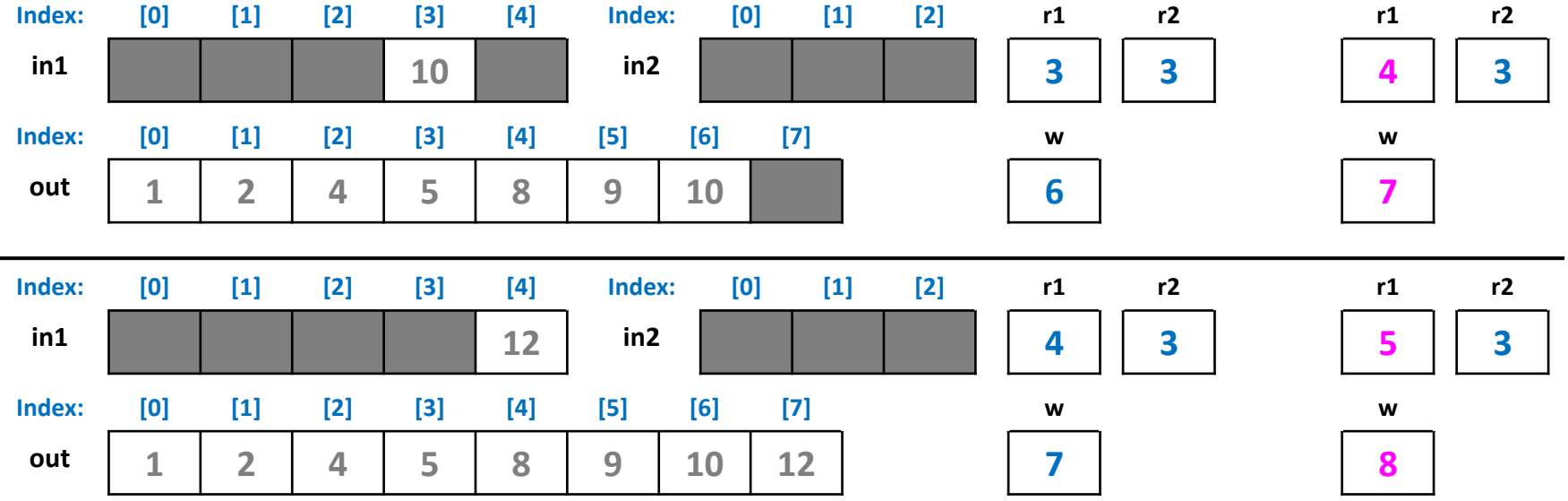
Task 8

If the user enters **3**, find **3** and return its index or -1:



Task 8

If the user enters **3**, find **3** and return its index or -1:



Task 8

- What programming issues (mechanics) should you think about?
  - What cases or "phases" exist for the merge process? What two elements should be compared and what element should be placed in the output?

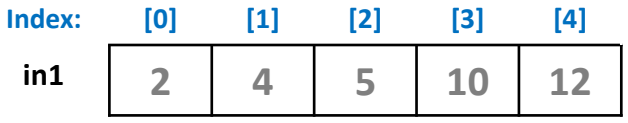
```
int main() {
  // setup array with data
  int n1=0, n2=0, in1[20], in2[20], out[40];
  int num;
  // Read array 1
  cin >> num;
  while(num != -1){
    in1[n1++] = num;
    cin >> num;
  }
  // Read array 2
  cin >> num;
  while(num != -1){
    in2[n2++] = num;
    cin >> num;
  }
  // See next column
```

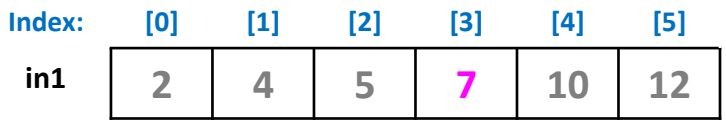
```
  // now perform the given task









  // Output the results
  return 0;
}
```

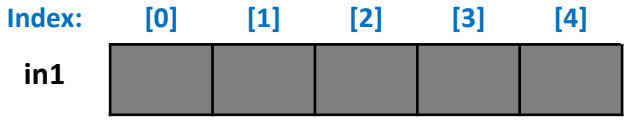
Task 9

- Given a SORTED array of length n, insert a value, val to a location that keeps the array sorted
- Questions:
  - How do we find the location to insert the value to?
  - What else do we have to do to avoid overwriting other values?

Insert 7 into the sorted array below

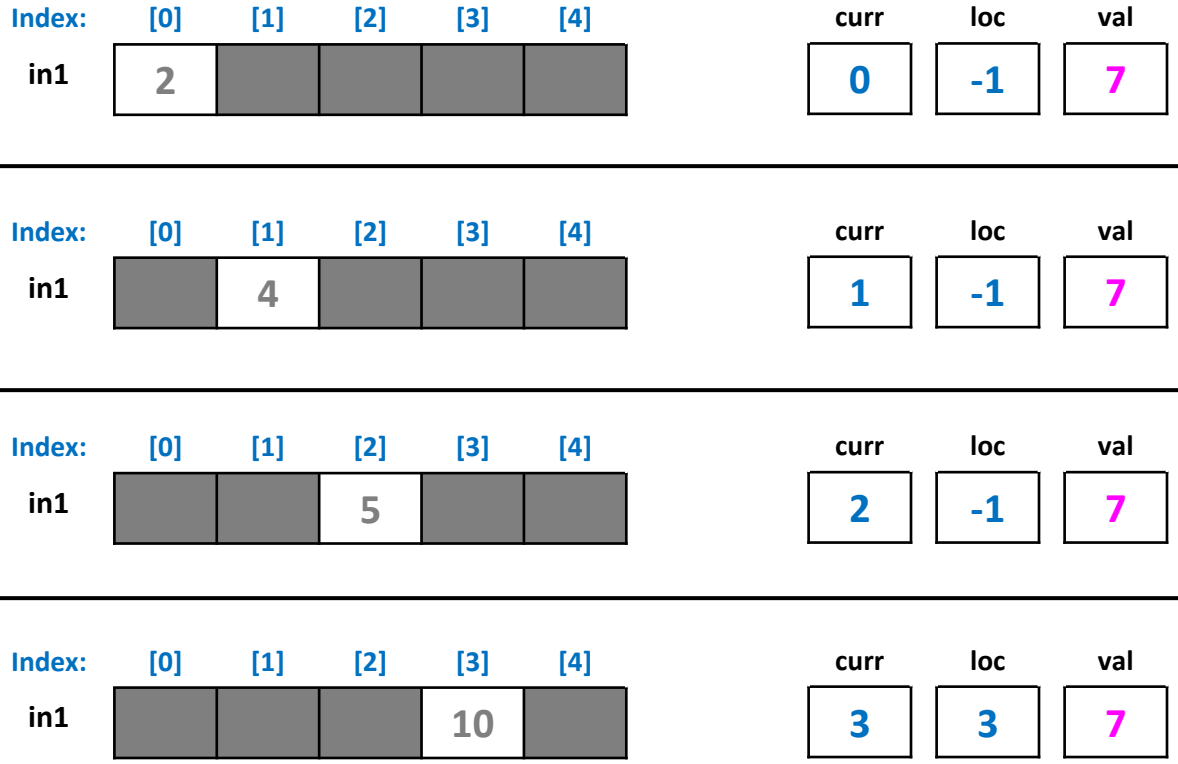






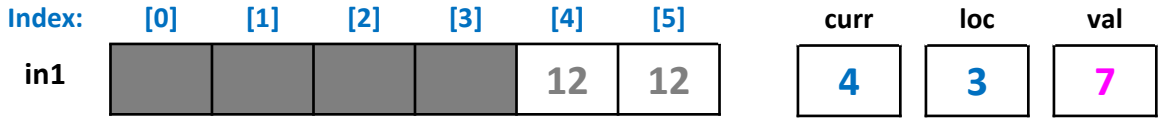
Task 9

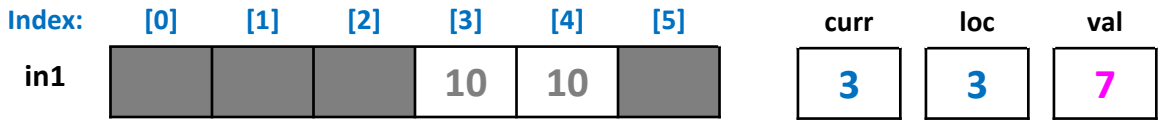
Insert 7 into the sorted array below

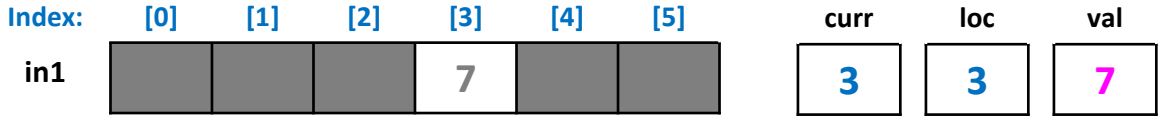


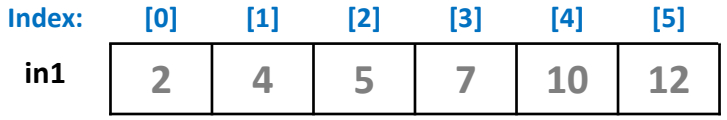
Task 9

Insert 7 into the sorted array below









Task 9

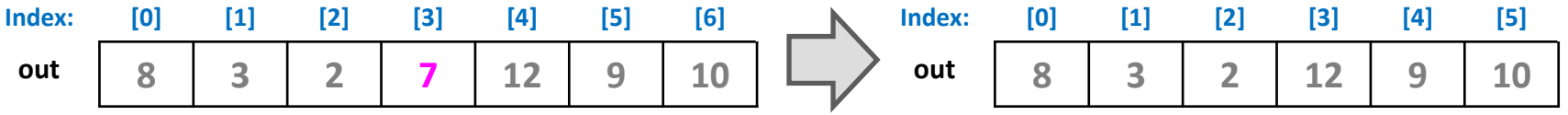
- What programming issues (mechanics) should you think about?
  - Are the indexes independent or is one dependent on another?
  - How and when are we ready to print our answer?
  - How do we stop (one or both loops)?

```
int main() {
  // setup array with data
  int n, data[100];
  cin >> n;
  for(int i=0; i < n; i++)
    { cin >> data[i]; }
  // now perform the given task









  // Output the results
  return 0;
}
```

Task 10

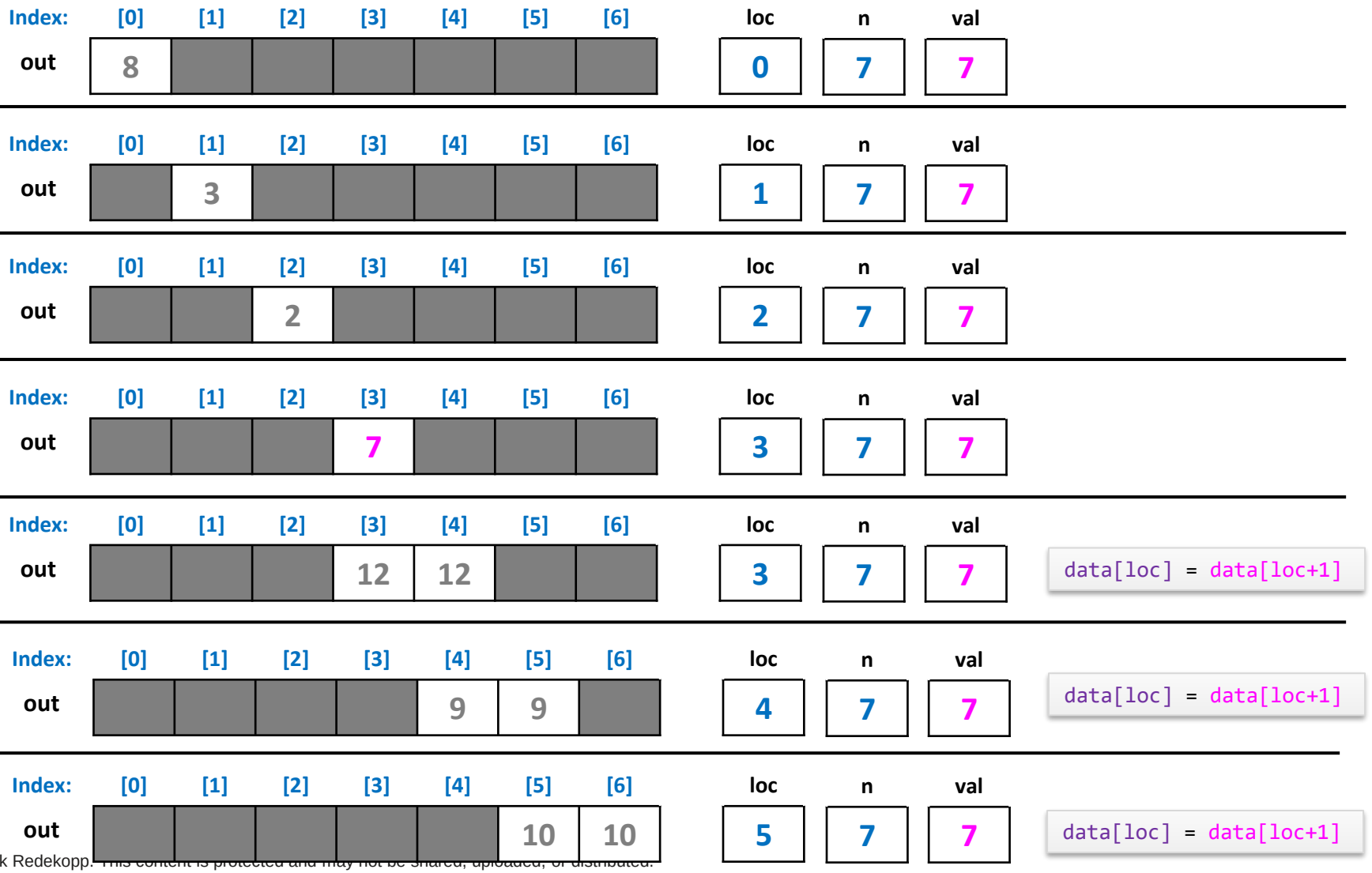
- Remove the first occurrence of a given value (if it exists) from an array, shifting values up
- Questions:
  - How can this be broken into 2 smaller tasks

Remove 7 from the array:



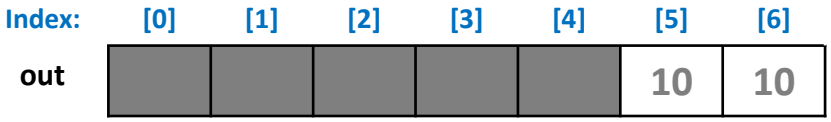
Task 10

Remove the first occurrence of val from the array



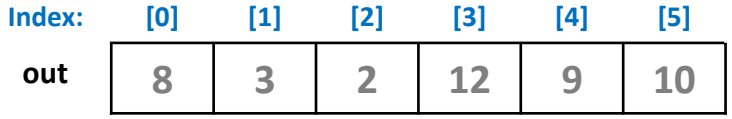
Task 10

Remove the first occurrence of val from the array



loc	n	val
5	7	7

```
data[loc] = data[loc+1]
```



loc	n	val
5	6	7

```
n--;
```

Task 10

- What programming issues (mechanics) should you think about?
  - In what order should you shift?
  - Will you shift from k to k-1 or k+1 to k? And where would you stop?

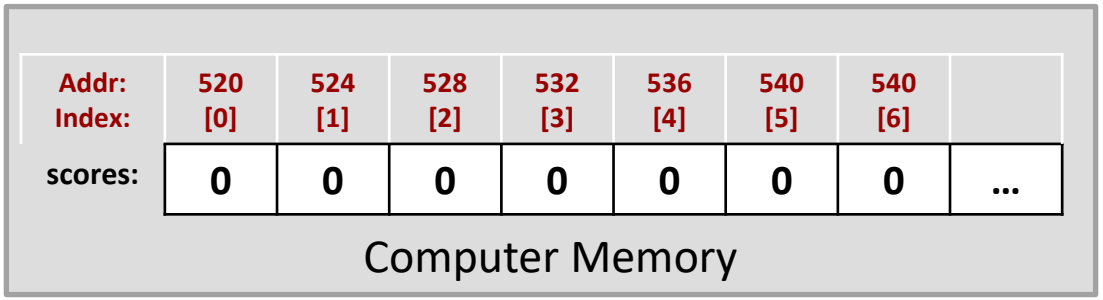
```
int main() {
  // setup array with data
  int n, data[100];
  cin >> n;
  for(int i=0; i < n; i++)
    { cin >> data[i]; }
  // now perform the given task












  // Output the results
  ...
  return 0;
}
```

INTERLUDE: ARRAY HOMEWORK

Sequential Iteration

- We usually iterate over an array sequentially, but this need not be the rule

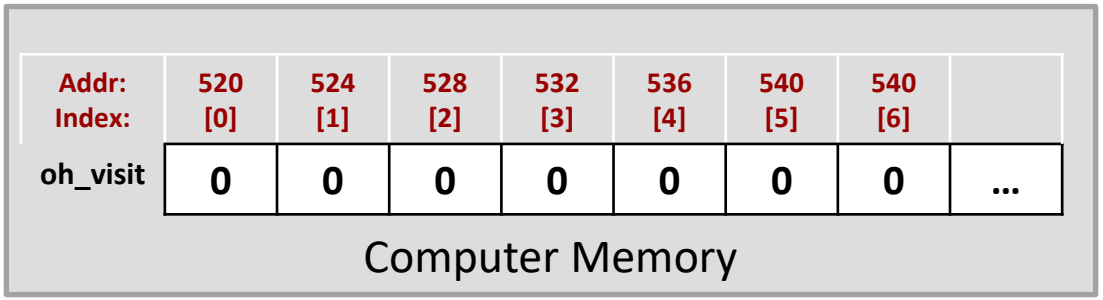
```
int main()
{
  int scores[100];
    // allocates 100 integers

  // initialize all to 0
  for(int i=0; i < 100; i++){
    scores[i] = 0;
  }

  // ..OR.. read in all entries
  for(int i=0; i < 100; i++){
    cin >> scores[i];
  }

}
```



Random Access (Indexing)

- We can access values in any random order.
- Suppose I say that any student that visits me in office hours will receive 2 additional points

```
int main()
{
  int oh_visit[100];
    // allocates 100 integers

  // loop to initialize array
  // to 0s

  int stu_id;
  cin >> stu_id;

  while( stu_id != -1) {

      oh_visit[stu_id] = 2;

      cin >> stud_id;

  }

}
```



Arrays as Look-Up Tables

- Use the value of one array as the index of another
- Suppose you are given some integers as data [in the range of 0 to 5]
- Suppose computing squares of integers was difficult (no built-in function for it)
- Could compute them yourself, record answer in another array and use data to “look-up” the square

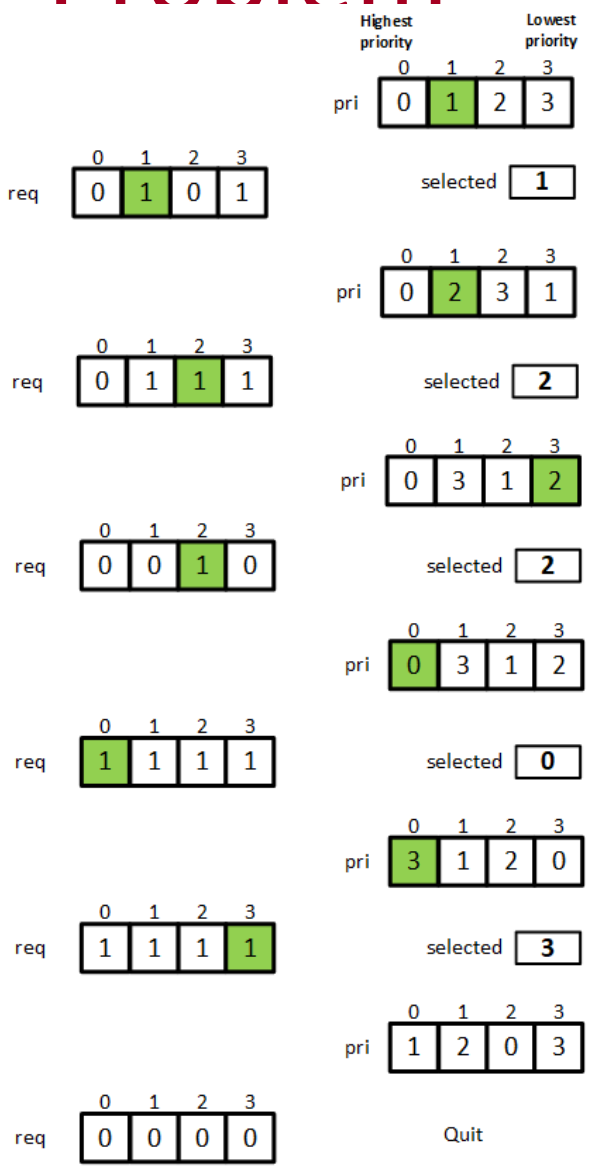
```
// the data
int data[8] = {3, 2, 0, 5, 1, 4, 5, 3};

// The LUT
int squares[6] = {0,1,4,9,16,25};
```

```
// the data
int data[8] = {3, 2, 0, 5, 1, 4, 5, 3};

// The LUT
int squares[6] = {0,1,4,9,16,25};

for(int i=0; i < 8; i++){
  int x = data[i]
  int x_sq = squares[x];
  cout << i << "," << x_sq << endl;
}
```

```
// the data
int data[8] = {3, 2, 0, 5, 1, 4, 5, 3};

// The LUT
int squares[6] = {0,1,4,9,16,25};

for(int i=0; i < 8; i++){
  int x_sq = squares[data[i]];
  cout << i << "," << x_sq << endl;
}
```

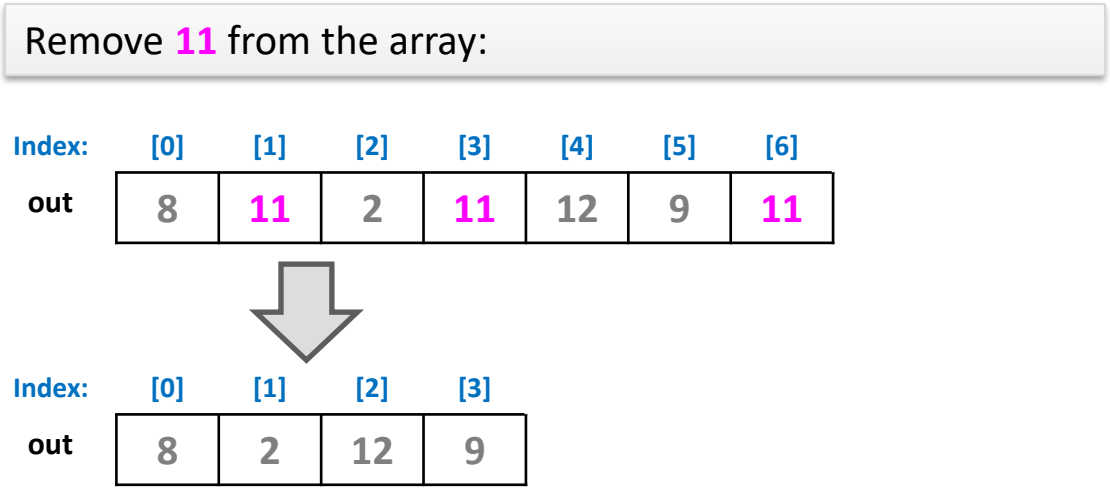
Approach to "Priority" Problem

- Find the 2D structure
- Consider when sequential loops are necessary vs. nesting loops
- In this program the *values* of the priority array can be used to check the requests array in order from most to least priority.



Task 11

- Remove ALL the occurrences of a given value from an array, shifting values up.
- Questions:
  - Can we do this in one pass? If so, what do we need to track?



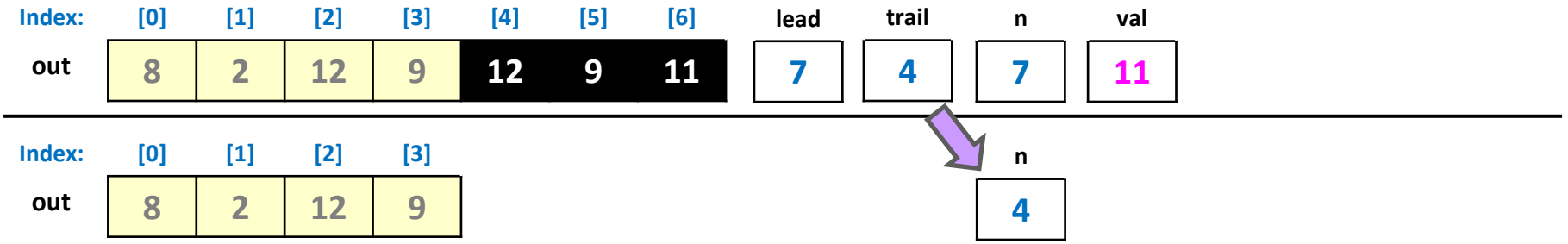
Task 11

Remove the first occurrence of val from the array

Index:	[0]	[1]	[2]	[3]	[4]	[5]	[6]	lead	trail	n	val	
out	8							0	0	7	11	`data[trail] = data[lead]` `trail++; lead++;`
Index:	[0]	[1]	[2]	[3]	[4]	[5]	[6]	lead	trail	n	val	
out	8	11						1	1	7	11	`lead++;`
Index:	[0]	[1]	[2]	[3]	[4]	[5]	[6]	lead	trail	n	val	
out	8	2	2					2	1	7	11	`data[trail] = data[lead]` `trail++; lead++;`
Index:	[0]	[1]	[2]	[3]	[4]	[5]	[6]	lead	trail	n	val	
out	8	2	2	11				3	2	7	11	`lead++;`
Index:	[0]	[1]	[2]	[3]	[4]	[5]	[6]	lead	trail	n	val	
out	8	2	12	11	12			4	2	7	11	`data[trail] = data[lead]` `trail++; lead++;`
Index:	[0]	[1]	[2]	[3]	[4]	[5]	[6]	lead	trail	n	val	
out	8	2	12	9	12	9		5	3	7	11	`data[trail] = data[lead]` `trail++; lead++;`
Index:	[0]	[1]	[2]	[3]	[4]	[5]	[6]	lead	trail	n	val	
out	8	2	12	9	12	9	11	6	4	7	11	`lead++;`

Task 11

Remove the first occurrence of val from the array



Task 11

- What programming issues (mechanics) should you think about?
  - Do we just need to track the maximum VALUE or the INDEX of the maximum value?
  - Given that you can move the maximum number to the end of the array, how could this be used to SORT the entire array?
    - Repeat the process for the first n-1 elements, then repeat for the first n-2 elements, etc.

```
int main() {
  // setup array with data
  int n, data[100];
  cin >> n;
  for(int i=0; i < n; i++)
    { cin >> data[i]; }
  // now perform the given task









  // Output the results
  for(int i=0; i < n; i++){
    cout << data[i] << " ";
  }
  cout << endl;
  return 0;
}
```

Task 12a

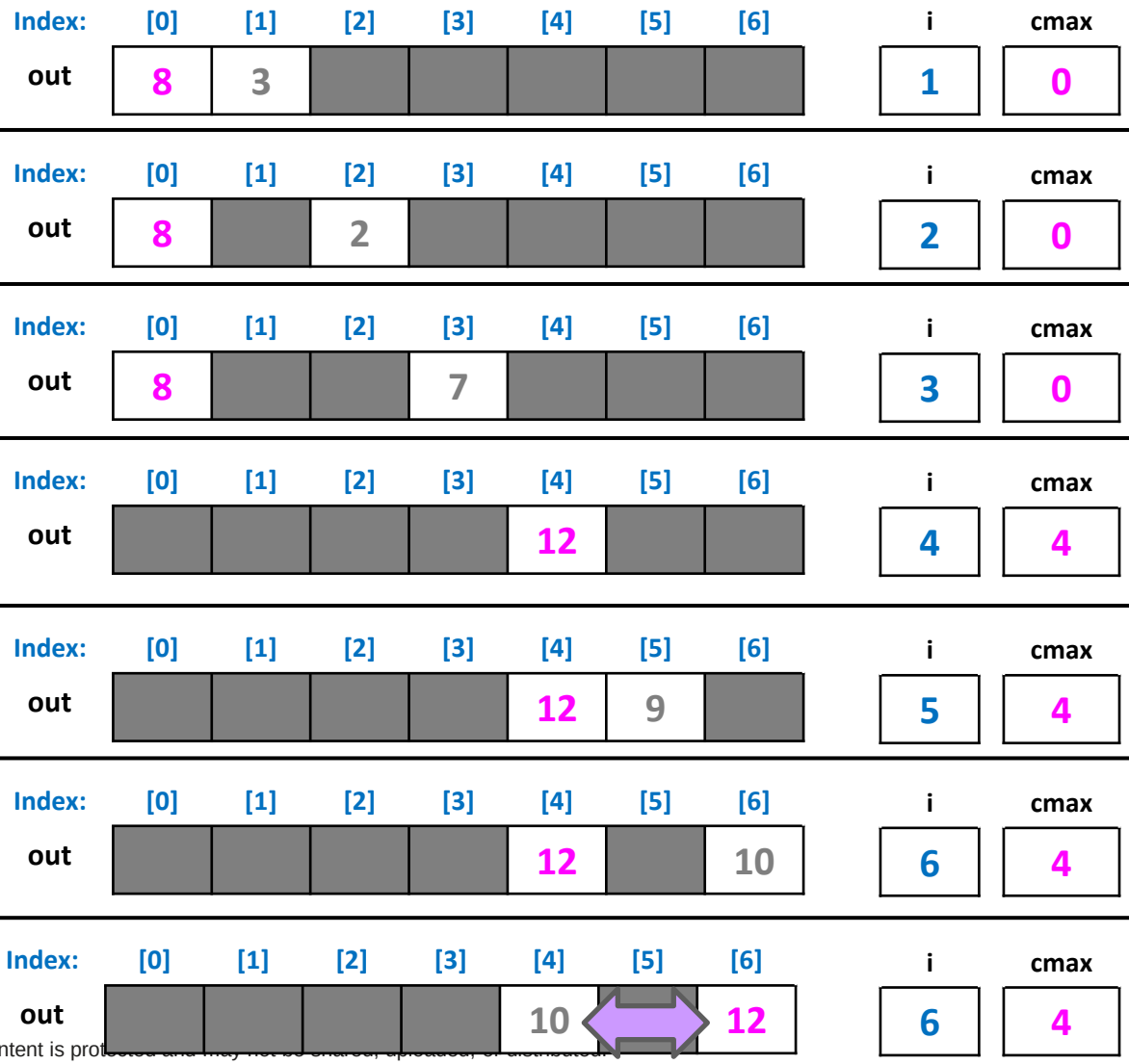
- Find the maximum value in an array and move it to the end of the array
- Questions:
  - Do we scan through the array to find the maximum without moving it and swap it at the end ..or..
  - Do we move it as we can through the array

Find the maximum value and move it to the end of the array.

Index:	[0]	[1]	[2]	[3]	[4]	[5]	[6]
out	8	3	2	7	12	9	10

Task 12a

Find the maximum value and move it to the end of the array.



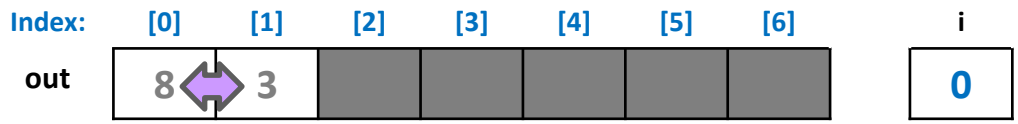
Task 12a

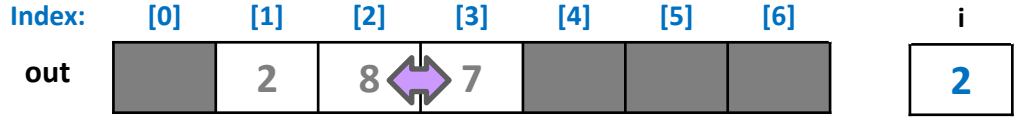
- What programming issues (mechanics) should you think about?
  - Do we just need to track the maximum VALUE or the INDEX of the maximum value?
  - Given that you can move the maximum number to the end of the array, how could this be used to SORT the entire array?

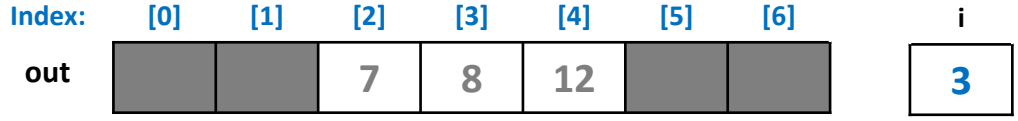
```
int main() {
  // setup array with data
  int n, val, data[100];
  cin >> n;
  for(int i=0; i < n; i++)
    { cin >> data[i]; }
  // now perform the given task









  // Print out results
  for(int i=0; i < n; i++){
    cout << data[i] << " ";
  }
  cout << endl;
  return 0;
}
```

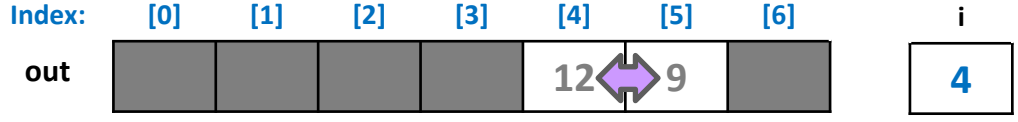
Task 12b

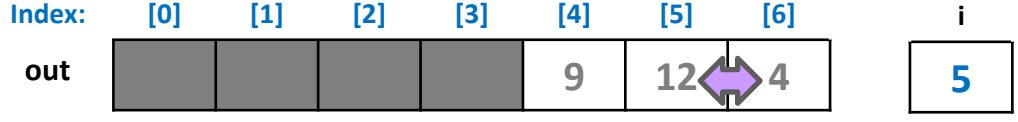
Find the maximum value and move it to the end of the array.

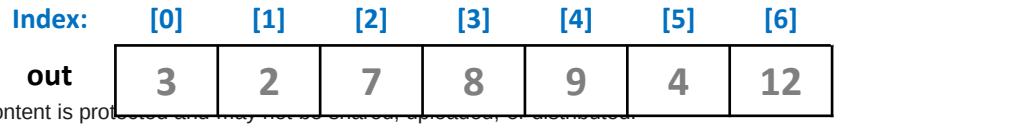












Task 12b

- What programming issues (mechanics) should you think about?
  - Do we just need to track the maximum VALUE or the INDEX of the maximum value?
  - Given that you can move the maximum number to the end of the array, how could this be used to SORT the entire array?

```
int main() {
  // setup array with data
  int n, val, data[100];
  cin >> n;
  for(int i=0; i < n; i++)
    { cin >> data[i]; }
  // now perform the given task









  // Print out results
  for(int i=0; i < n; i++){
    cout << data[i] << " ";
  }
  cout << endl;
  return 0;
}
```

SOLUTIONS

Task 8 - Sol

- What programming issues (mechanics) should you think about?
  - What cases or "phases" exist for the merge process? What two elements should be compared and what element should be placed in the output?

```
int main() {
  // setup array with data
  int n1=0, n2=0, in1[20], in2[20], out[40];
  int num;
  // Read array 1
  cin >> num;
  while(num != -1){
    in1[n1++] = num;
    cin >> num;
  }
  // Read array 2
  cin >> num;
  while(num != -1){
    in2[n2++] = num;
    cin >> num;
  }
  // See next column
```

```
  // now perform the given task
  int r1 = 0, r2 = 0, w = 0;
  while(r1 < n1 && r2 < n2) {
    if(in1[r1] < in2[r2]) {
      out[w++] = in1[r1++];
    }
    else {
      out[w++] = in2[r2++];
    }
  }
  while(r1 < n1) { // place remaining in1
    out[w++] = in1[r1++];
  }
  while(r2 < n2) { // place remaining in2
    out[w++] = in2[r2++];
  }
  // Output the results
  return 0;
}
```

Task 9 - Sol

- What programming issues (mechanics) should you think about?
  - Are the indexes independent or is one dependent on another?
  - How and when are we ready to print our answer?
  - How do we stop (one or both loops)?

```
int main() {
  // setup array with data
  int n, data[100];
  cin >> n;
  for(int i=0; i < n; i++)
    { cin >> data[i]; }
  // now perform the given task
  int val, loc = -1, curr = 0;
  cin >> val;
  if(n < 100){
     while(curr < n && val > data[curr] ) {
      curr++;
    }
    loc = curr;
    for(int curr = n-1; curr >= loc; curr--) {
      data[curr+1] = data[curr];
    }
    data[loc] = val;
    n++;
  }
  else {
    cout << "No room" << endl;
  }
  // Output the results
  return 0;
}
```

Task 10 - Sol

- What programming issues (mechanics) should you think about?
  - In what order should you shift?
  - Will you shift from k to k-1 or k+1 to k? And where would you stop?

```
int main() {
  // setup array with data
  int n, data[100];
  cin >> n;
  for(int i=0; i < n; i++)
    { cin >> data[i]; }
  // now perform the given task
  int val, loc;
  cin >> val;
  // find first occurrence of val
  for(loc = 0; loc < n; loc++) {
    if(data[loc] == val) { break; }
  }
  if(loc < n) {
    // shift items up from loc to n
    //  invariant: data[loc] is always safe
    //             to overwrite
    for( ; loc < n-1; loc++) {
      data[loc] = data[loc+1];
    }
    n--;
  }
  // Output the results
  ...
  return 0;
}
```

Task 11 - Sol

- What programming issues (mechanics) should you think about?
  - Do we just need to track the maximum VALUE or the INDEX of the maximum value?
  - Given that you can move the maximum number to the end of the array, how could this be used to SORT the entire array?
    - Repeat the process for the first n-1 elements, then repeat for the first n-2 elements, etc.

```
int main() {
  // setup array with data
  int n, data[100];
  cin >> n;
  for(int i=0; i < n; i++)
    { cin >> data[i]; }
  // now perform the given task
  int val, lead, trail;
  cin >> val;
  trail = 0;
  for(lead = 0; lead < n; lead++) {
    if(data[lead] != val) {
      data[trail] = data[lead];
      trail++;
    }
  }
  n = trail;
  // Output the results
  for(int i=0; i < n; i++){
    cout << data[i] << " ";
  }
  cout << endl;
  return 0;
}
```

Task 12a - Sol

- What programming issues (mechanics) should you think about?
  - Do we just need to track the maximum VALUE or the INDEX of the maximum value?
  - Given that you can move the maximum number to the end of the array, how could this be used to SORT the entire array?
    - Repeat the process for the first n-1 elements, then repeat for the first n-2 elements, etc.

```
int main() {
  // setup array with data
  int n, val, data[100];
  cin >> n;
  for(int i=0; i < n; i++)
    { cin >> data[i]; }
  // now perform the given task
  int cmax = 0;
  for(int i=1; i < n; i++) {
    if(data[i] > data[cmax]){
       cmax = i;
    }
  }
  // swap the max and end element
  int temp = data[n-1];
  data[n-1] = data[cmax];
  data[cmax] = temp;
  // Print out results
  for(int i=0; i < n; i++){
    cout << data[i] << " ";
  }
  cout << endl;
  return 0;
}
```

Task 12b - Sol

- What programming issues (mechanics) should you think about?
  - Do we just need to track the maximum VALUE or the INDEX of the maximum value?
  - Given that you can move the maximum number to the end of the array, how could this be used to SORT the entire array?
    - Repeat the process for the first n-1 elements, then repeat for the first n-2 elements, etc.

```
int main() {
  // setup array with data
  int n, val, data[100];
  cin >> n;
  for(int i=0; i < n; i++)
    { cin >> data[i]; }
  // now perform the given task
  for(int i=0; i < n-1; i++) {
    if(data[i] > data[i+1]){
       int temp = data[i];
       data[i] = data[i+1];
       data[i+1] = temp;
    }
  }
  // Print out results
  for(int i=0; i < n; i++){
    cout << data[i] << " ";
  }
  cout << endl;
  return 0;
}
```

Task 13 - Partition

- TBD