COSC 2436: Exam 1 Review

J. Eoin Donovan - 2023

Linked Lists #1

What is wrong with the linked list function below?

```
node *removeTail(node *head){
  node *prev = nullptr;
  node *cur = head;
  while(cur != nullptr){
    cur = cur->next;
    prev = cur;
  }
  prev->next = nullptr;
  delete cur;
}
```

Linked Lists #1

What is wrong with the linked list function below?

```
node *removeTail(node *head){
  node *prev = nullptr; //We need to check if list is empty first
  node *cur = head;
  while(cur != nullptr){ //Should be cur->next != nullptr
    cur = cur->next; //We need to set prev = cur before
    prev = cur;      //cur = cur->next
  }
  prev->next = nullptr; //This will be a seg fault if head is NULL
  delete cur;
} //Doesn't return the head at the end
```

Linked List #2

What are some advantages of using a linked list and what are some disadvantages?

Linked List #2

What are some advantages of using a linked list and what are some disadvantages?

Some advantages of linked list include size can dynamically grow and shrink, efficient insertion and deletion (no shifting), and does not require contiguous memory allocation.
Some disadvantages of linked list include no random access, more memory usage than array, traversal can sometimes be more time consuming.

Linked List #3

What is the role of the head and tail pointers in a linked list?

Linked List #3

What is the role of the head and tail pointers in a linked list?

The head pointer keeps track of the first node in a linked list and the tail pointer keeps track of the last node in a linked list.

Linked List #4

Complete the function below which prints the 2nd from the end value in a linked list. If the list is empty or has only 1 node, print nothing.

```
void print2ndfromEnd(node *head){





}
```

Linked List #4

```
void print2ndfromEnd(node *head){
  if(head == nullptr || head->next == nullptr){
    return;
  }
  node *cur = head;
  while(cur->next->next != nullptr){
    cur = cur->next;
  }
  cout << cur->value << endl;
}
```

Recursion #1

Write a recursive function to find the sum of the first n natural numbers. Example: Given 4, the function would return 10 since 1+2+3+4 = 10.

```
int getSum(int n){



}
```

Recursion #1

Write a recursive function to find the sum of the first n natural numbers. Example: Given 4, the function would return 10 since 1+2+3+4 = 10.

```
int getSum(int n){
  if(n == 1)
    return n;
  return n + getSum(n-1);
}
```

Recursion #2

What is the output of fun(5)?

```
void fun(int n){
  if(n <= 1){
    return;
  }
  else if(n == 2 || n == 3){
    cout << n << endl;
    fun(n-1);
  }
  else{
    fun(--n);
    cout << n << endl;
  }
}
```

Recursion #2

What is the output of fun(5)?

```
void fun(int n){
  if(n <= 1){
    return;
  }
  else if(n == 2 || n == 3){
    cout << n << endl;
    fun(n-1);
  }
  else{
    fun(--n);
    cout << n << endl;
  }
}
```

```
3
2
3
4
```

`fun(--n) will decrement the value of n in memory while fun(n-1) will not`

Sorting #1

Sort the array below using selection sort. Show the array after every time a swap takes place.

```
{6, 3, 4, 9, 1, 2}
```

Sorting #1

Sort the array below using selection sort. Show the array after every time a swap takes place.

```
{6, 3, 4, 9, 1, 2}
{1, 3, 4, 9, 6, 2}
{1, 2, 4, 9, 6, 3}
{1, 2, 3, 9, 6, 4}
{1, 2, 3, 4, 6, 9}
```

Sorting #2

Give the best case and worst case time complexities for the floowing sorts.

Bubble Sort (Not Enhanced):

Best Case: **Worst Case:**

Insertion Sort:

Best Case: **Worst Case:**

Selection Sort:

Best Case: **Worst Case:**

Sorting #2

Give the best case and worst case time complexities for the floowing sorts.

Bubble Sort (Not Enhanced):
Best Case: $O(n^2)$ Worst Case: $O(n^2)$

Insertion Sort:
Best Case: $O(n)$ Worst Case: $O(n^2)$

Selection Sort:
Best Case: $O(n^2)$ Worst Case: $O(n^2)$

Sorting #3

Explain how selection sort works.

Sorting #3

Explain how selection sort works.

Selection sort works by repeatedly selecting the smallest (or largest of sorting in descending order) element from the unsorted portion of the list and moving it to the sorted portion of the list.

Array #1

What are some advantages and disadvantages of arrays?

Array #1

What are some advantages and disadvantages of arrays?

Some advantages of array include random access, easy iteration, and create multi-dimensional arrays.
Some disadvantages of array include fixed size, memory wastage, and shifting when inserting/deleting in the middle.

Time Complexity #1

What is the time complexity of the code shown below?

```
for(int i = n; i > 0; i /= 2){
  for(int j = n; j > 0; j /= 2){
    cout << i + j << endl;
  }
}
```

Time Complexity #1

What is the time complexity of the code shown below?

```
for(int i = n; i > 0; i /= 2){
 for(int j = n; j > 0; j /= 2){
    cout << i + j << endl;
  }
}
```

$O((\log n)^2)$

Time Complexity #1

What is the time complexity of the code shown below?

```
for(int i = n; i > 0; i--){
  for(int j = 0; j < n; j++){
    cout << i + j << endl;
  }
}
```

Time Complexity #1

What is the time complexity of the code shown below?

```
for(int i = n; i > 0; i--){
  for(int j = 0; j < n; j++){
    cout << i + j << endl;
  }
}
```

$O(n^2)$

Exam 1 Information

- The exam is on paper
- Pencils will <u>not</u> be provided so you will need to bring your own pencils and/or erasers
- You will have 75 minutes to take the exam
- For study material, you can review the powerpoints and code that are posted on the Canvas page for your lab section