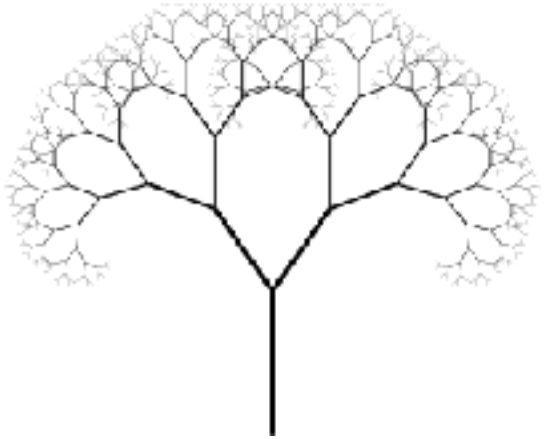
STRINGS
RECURSION

Problem Solving with Computers-I



Announcements

Final review in Phelps 3526 on Friday (06/08) from 3p - 5p

Strings

Q1: How are ordinary arrays of characters and C-strings similar and how are they dissimilar?

The C++ string class methods

```
string fruit = "Apple";
int len = fruit.length();
int pos= fruit.find('l');
string part= fruit.substr(1,3);
fruit.erase(2,3);
fruit.insert(2,"ricot");
fruit.replace(2,5,"ple");

Check out cctype for checks and conversions on characters
fruit[0]= tolower(fruit[0]);
isalpha(fruit[0])
isalnum(fruit[0])
```

What is the output of the code?

```
char s1[] = "Mark";
char s2[] = "Jill";
for (int i = 0; i <= 4; i++)
      s2[i] = s1[i];
if (s1 == s2) s1 = "Art";
cout<<s1<<" "<<s2<<endl;
```

A. Mark Jill
B. Mark Mark
C. Art Mark
D. Compiler error
E. Run-time error

What is the output of the code?

```
string s1 = "Mark";
string s2 = "Jill";
for (int i = 0; i <= s1.length(); i++)
      s2[i] = s1[i];
if (s1 == s2) s1 = "Art";
cout<<s1<<" "<<s2<<endl;
```

A. Mark Jill
B. Mark Mark
C. Art Mark
D. Compiler error
E. Run-time error

Lab 08: anagrams and palindromes

```
bool isPalindrome(string s1)
```

deTartraTED
WasItACarOrACatISaw

```
bool isAnagram(string s1, string s2)
```

Diba == Adib
Rats and Mice == In cat's dream
Waitress == A stew, Sir?

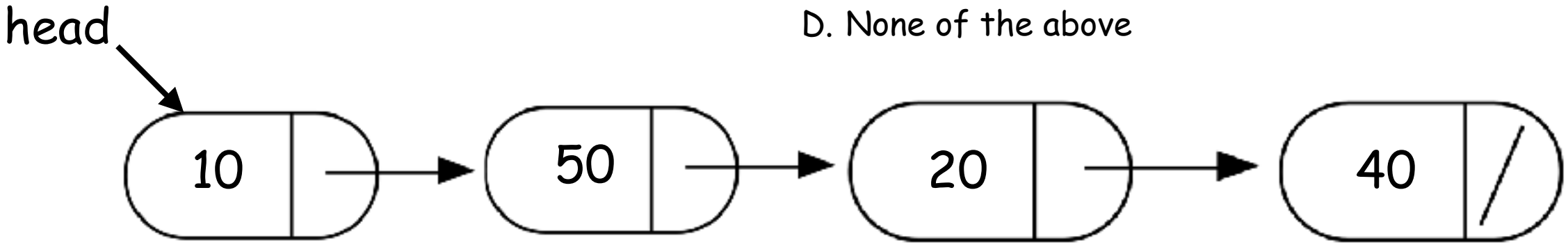
Why don't we pass the length of the string?

```
HW-9, Q7
//return the sum of all the elements in a linked list
double sumList(Node* head){









}
```

What's in a base case?

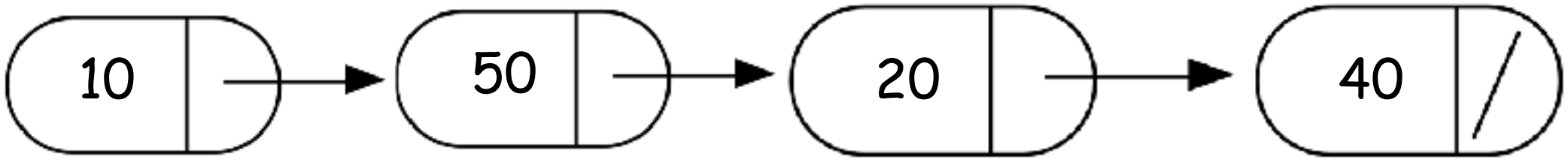
What happens when we execute this code on the example linked list?

A. Returns the correct sum (120)
B. Program crashes with a segmentation fault
C. Program runs forever
D. None of the above

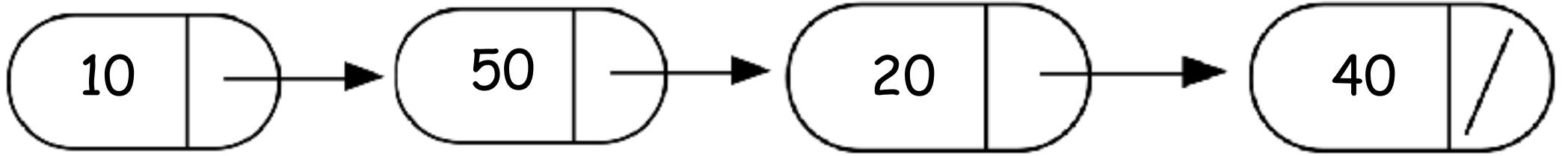



```
double sumList(Node* head){

    double sumRest;
    sumRest = sumList(head->next);
    return head->data + sumRest;
}
```

```
double sumList(Node* head){

    double sumRest;
    sumRest = sumList(head->next);
    return head->data + sumRest;
}
```



```
double sumList(Node* head){


     double sumRest;
    sumRest = sumList(head->next);
    return head->data + sumRest;
}
```



Helper functions

- Sometimes your functions takes an input that is not easy to recurse on
- In that case define a new function with appropriate parameters: This is your helper function
- Call the helper function to perform the recursion

For example

```
double sumLinkedList(LinkedList* list){
    return sumList(list->head); //sumList is the helper
    //function that performs the recursion.

}
```

Next time

- Advanced problems with recursion
- Final review