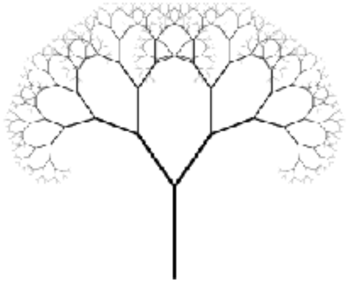
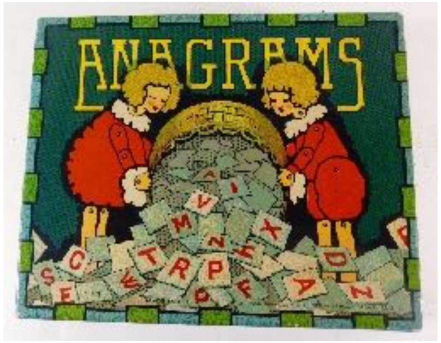
STRINGS RECURSION

Problem Solving with Computers-I





Announcements

1. Final review in Phelps 352~~6~~5 on Friday (06/08) from 3p - 5p
2. Information about the final exam, seating chart and practice problems are available at:

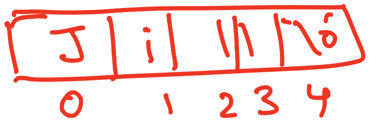
https://ucsb-cs16-s18-mirza.github.io/exam/e03/

Strings

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

Char array : char arr[] = { 'J', 'i', 'l', 'l' };
on
array of chars

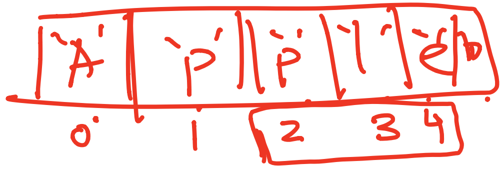
C-string : char arr[] = { 'J', 'i', 'l', 'l', '\0' };
OR
Char arr[] = " Jill " ;



arr[0] 'J'

The C++ string class methods

fruit



fruit[0]

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

cctype

Check out cctype for checks and conversions on characters

```
fruit[0]= tolower(fruit[0]);
isalpha(fruit[0])
isalnum(fruit[0])
```

convert to lowercase

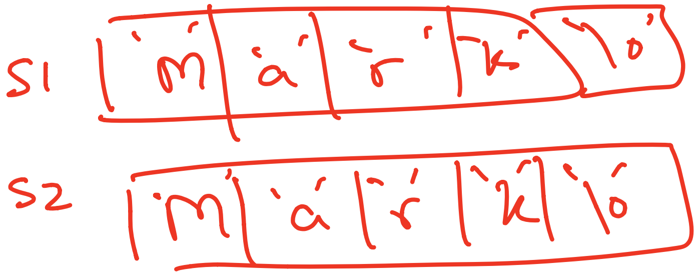
for (int i = 0 ; i< fruit.length(); i++){

fruit.erase (i, 2);

True if fruit[0] is an alphabet or number?

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;
```



comparing &s1[0] and &s2[0] (These can never be equal)

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

s1 = "Art"; cannot change the starting location of an array

If we replace s1="Art"; by cout<<"Art"; then the output will be Mark Mark, why?

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;
```

4

same as

$\rightarrow$ s2 = s1;

A. Mark Jill

B. Mark Mark

[✓ C.] Art Mark

D. Compiler error

E. Run-time error

Lab 08: anagrams and palindromes

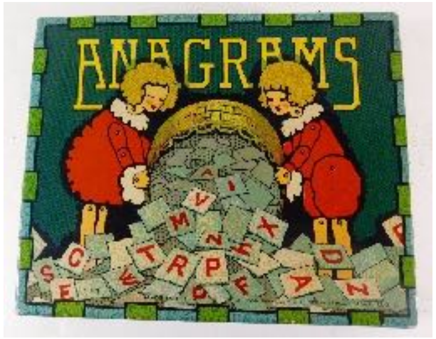
```
bool isPalindrome(string s1)
```

deTartraTED $\longrightarrow$ $\underrightarrow{\text{detartrated}}$ ape apa
WasItACarOrACatISaw

$\xleftarrow{\hspace{3cm}}$

```
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?

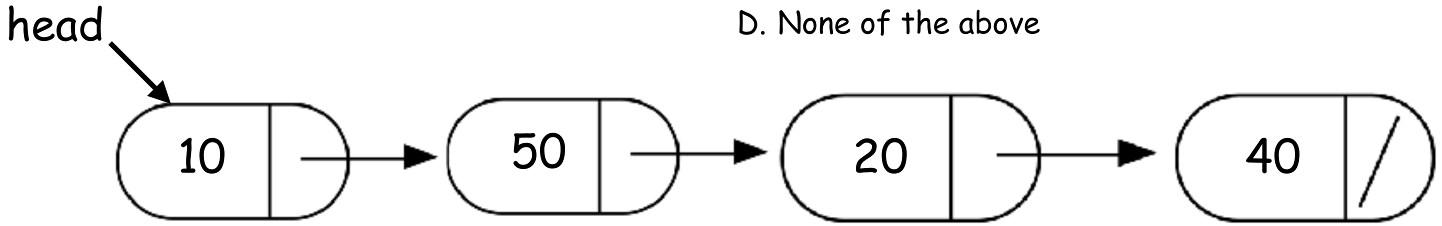
HW-9, Q7

```
//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

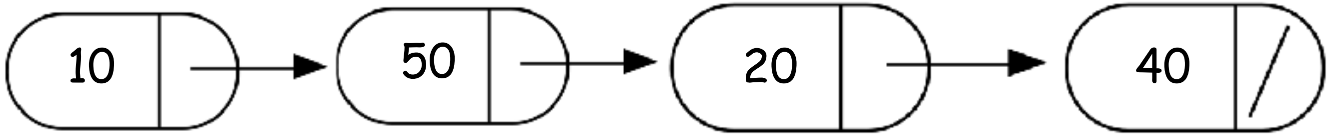


```
double sumList(Node* head){

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

```
double sumList(Node* head){

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

head

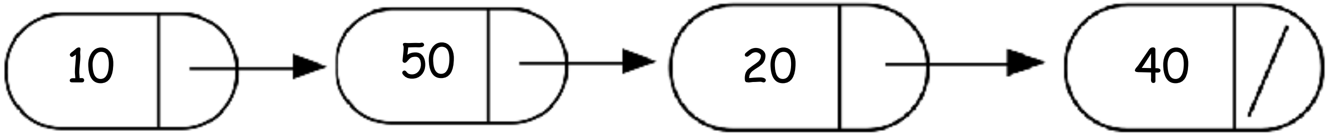
sumRest



head

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