Session Name: 26 - Review 4 5-5-2022 12-58 PM

Date Created: 5/5/22, 12:29:37 PM **Active Participants:** 91 of 111
Average Score: 0.00% **Questions:** 2

Results by Question

1. Please do question 1 which is on the screen in class. (Short Answer)

	Responses	
	Percent	**Count**
619 983 036 783 734 186 494 469 251	17.98%	16
186 251 036 469 494 619 734 783 983	6.74%	6
619 983 036 783 734 186 494 469 619	5.62%	5
036 186 251 469 494 619 734 783 983	4.49%	4
251 036 186 494 469 619 983 783 734	4.49%	4
251 983 036 783 619 186 494 469 734	4.49%	4
619	3.37%	3
036 186 251 469 494 734 783 983	2.25%	2
251 983 036	2.25%	2
619 251 983 036 783 734 186 494 469	2.25%	2
036 186 251 469 494 619 734 783	1.12%	1
036 186 251 983 783 734 494 469 619	1.12%	1
036 186 494 469 251 619 983 783 734	1.12%	1
036 186 494 469 619 983 783 734	1.12%	1
036 251 783 983	1.12%	1
036 251 783 983 186 469 494 619 734	1.12%	1
186 036 469 494 619 734 783 983	1.12%	1
251	1.12%	1
251 036 186 469 494 619 734 783	1.12%	1
251 036 469 494186 734 783 983	1.12%	1
251 469 036 494 186 619 783 983 734	1.12%	1
251 469 036 494 186 734 783 983 619	1.12%	1
251 469 036 494 619 734 783 983	1.12%	1
251 469 036 496 186 619 734 783 983	1.12%	1

251 619 734	1.12%	1
251 734 619	1.12%	1
251 983 036 619 783 186 494 469 734	1.12%	1
251 983 036 783	1.12%	1
251 983 036 783 619	1.12%	1
251 983 036 783 619 734 186 494 496 734	1.12%	1
251, 036, 186, 494, 469, 619, 983, 783,734	1.12%	1
251,469,036,494,186,619,734,763,983	1.12%	1
251036469186 469619783983	1.12%	1
256 469 036 494 186 619 734 783 983	1.12%	1
469 494 619	1.12%	1
619 036 251 734 783 186 469 494 619	1.12%	1
619 251	1.12%	1
619 251 036 186 734 783 494 469 251	1.12%	1
619 251 469	1.12%	1
619 251 469 036 783 734 186 494 983	1.12%	1
619 251 734 983 036 783 186 494 469	1.12%	1
619 469 036 494 186 734 783 983 751	1.12%	1
619 983 036 734 734 186 494 469 251	1.12%	1
619 983 036 783 734 186 494 169 251	1.12%	1
619, 983, 036, 783, 734, 186, 494, 469, 251	1.12%	1
619251469036494 186784783983	1.12%	1
691 251 036 469 494 186 734 783 983	1.12%	1
734 251 036 469 186 619 783 494 983	1.12%	1
783	1.12%	1
983 734 469	1.12%	1
I DON'T KNOW	1.12%	1
Totals	**100%**	**89**

Keyword(s): 186, 251, 036, 469, 494, 619, 734, 783, 983

Keyword Matches: 0

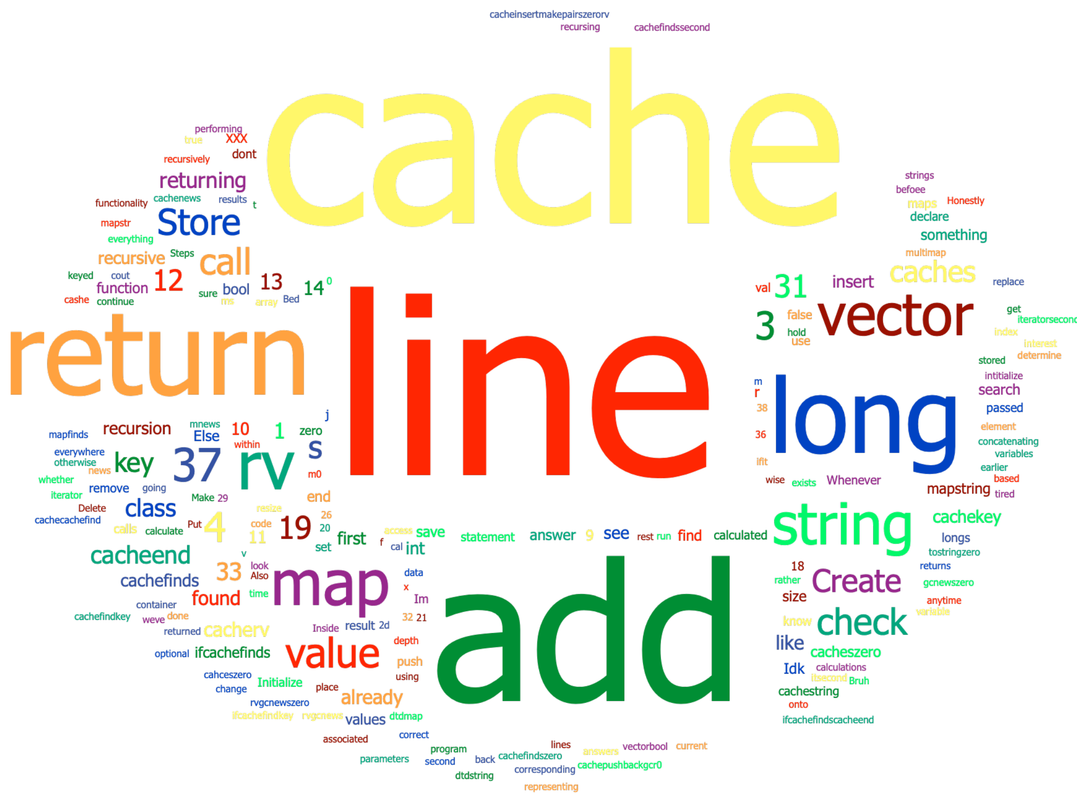
2. Please do question 2 which is on the screen in class. (Essay)

Responses
1
2d map with string as key and bool as key of second with val of rv
Add a cache to class variables Create cache if it's the first call Return answer from cache if it's there If not do recursion Put the answer into the cache before returning
Add a cache to the class "map<string, int> cache" Before like 12 add "string key = s + to_string(zero); if(cache.find(key) != cache.end) return cache[key]" before return on line 37 add "cache[key] = rv^gc(news,zero);"
Add a cache within the XXX class Whenever the program is going to return, search the cache first before. Then, whenever you recursively call the function, cache the value

Add a cache, vector or map of return values of interest
Add cache and return cache if found. Before returning anytime, insert into cache
Add cache functionality Store in cache when returning Check whether value exists in cache before performing calculations
add cache map after line 3, store by key string, val is long long
Add cache to class In line 31, check if r, false is in cache Before returning in line 37 check cache for news, zero
add cashe everywhere you call recursion, search in cache first
add int j before line 9
Add vector of long longs as cache after line 3, store rv in cache after line 18, return from cache at the end
after line 10, add vector <long long> v after line 29, push back results into vector and call vector returns before like 31
after line 10, declare a cache before line 12, check the cache for the value line 19, save rv in cache before you return line 33 save rv in cache befoee returning line 37 save in cache before returning rv
after line 3 add a cache, map <string, long long> cache; after line 20 add, if(cache.find(s) != cache.end()) return cache.find(s); after line 31 add, cache[s] = rv;
After line 3 add map <string, long long> cache; After line 13 add string key = s; if (cache.find[key] != cache.end()) return cache[key]; After line 36 add cache[key] r rv ^gc(news,zero);
after line 3, add "map <string, long long> cache;" after line 12, add "if (cache.find(s+zero) != cache.end()) return cahce[s+zero]" before line 33, add "cache.insert(make_pair(s+zero,rv))"
After line 3, add map for cache; after line 10, check is string is in map - if so return the value, otherwise continue
After line 3, create a map for the cache. Before line 12, calculate the key based on concatenating the string with 0 for false bool or 1 for true bool. Also, if the key is found in the cache, return its stored data. Before line 37, store the return value in the cache at the key that was calculated earlier.
After line 38, add "cout << "no" << end;"
after line something add something
Bed Before line 37, add cache
Before line 11, add “vector <long long> cache;” Before line 32, add “if (cache[rv] != -1) return cache[rv];”
Before line 12 add it = cache.find(s) "if(it != cache.end()) return it
Before line 12, add "if (cache[s][zero] != -1) return cache[s][zero];" before line 33, add "cache[s][zero] = rv;"
Before line 12, create vector of long long, resize it to size of S with -1 if cache is not -1, return cache before return rv, store rv into cache vector and return cache vector on line 37,

store it to cache and return cache vector
Before line 12, initialize a vector of long longs Before lines 19 and 33, store rv in the vector cache
Before line 19, add cache
Before Line 19, multimap cache add rv at index of s size Before line 31, check if in cache before return call
before line 19, store rv in cache
Before line 26, add r = s
Before line 3 add map<str, long long> cache Before line 14 add if cache.find(s) return cache.find(s)->second Delete line 37 and replace with cache[s] = rv Idk what else
before line 3, add a container representing the cache; before line 19, insert rv into cache; line 31, access cache rather than recursive call
Before line 3, create a vector of strings for the cache. Before line 11, check the cache to see if we've done that string before, if we have return it. Other wise run the rest of the code and place the result into the cache
before line 31 add something i dont know im very tired
before line 4 add "vector <long long> cache line 13 intitialize cache
before line 4 add vector cache
Before line 4, add "map <long long, s> cache" Before line 19, add "cache[rv] = s" Inside the if statement on line 33, add "cache[rv] = s" Before line 14, add an if statement to call find on the cache and return if found
before line 4, add "map <string, long long> cache"
Before line 4, add "map <string, long long> cache; Before 19, add cache[rv
Before line 4, add "vector <string>;"
before line 4, add "vector<bool> cache
before line 4, add a cache map before line 13, check if in map before line 37, add it into the map
Before line 4, add cache array. Before line 14, determine if passed string already in cache. If so, return the corresponding value. Before line 37, add passed string with its return value to cache.
Before line 4, add "vector <int> cache" On line 31, change to "cache.push_back(gc(r,0))"
Before line 4, create a map<string, long long> to hold the cache. On line 11, call find on the cache on the current function call. If return of find is not the end iterator, return iterator->second. Before line 37, store the return value in the cache.
before line 9, add "map <string, long long> m;" before line 19, add "m[s] = rv;" In line 33 before return add "m["0"] = rv;" In line 37 before return add "m[news] = rv;"
Before line 9, add int j. Before line 12, add "if map.find(s), return map[s]; Before line 4, add "dtd::map <dtd::string, long long> map" Before line 37, add "map[s] = rv^gc(news, zero);"

Below line 3 add “vector <string> t” Below that add “vector <string> f” Store
Bruh
Cache answers. Steps 3 and 4 are optional
create a cache in the XXX class, add each result to cache, and add a check to see if we already have the answer in cache before recursing
Create a cache variable on line 10. On line 19, store rv into cache and then when using recursion, check to see if values are already in cache.
Create a map in class Store values returned by recursive calls in map Before recursive calls check if it is in map
Create cache vector If s in cache return cache[s] Set cache[s] = s
Declare a map keyed on the parameters as the cache Initialize the cache's size If a call already has an associated element in the cache, return that value Else, create it and return the correct value
Have map as cache. At line 13, add "if cache.find(s)" and return it's value if found. Before like 37, add cache[news] = rv
Honestly idk
I don’t know :(
Idk
Insert before line 4, map<string, long long> cache; Insert at line 21, if(cache.find(s) != cache.end) return it->second
I’m not sure
Like 13: check a cache to see if that value has already been calculated. Any other time we call return, push that value onto a cache
line 11, add map cache of string and long long before line 14, if string is found in cache, return long long at cache[string] before like 37, add long long at string to cache[string]
Line 31 look for everything value in cache else cal the function
Line 37, return rv and remove recursive call Before line 3, add cache Line 31, remove recursion Line 31, use cache to set rv
Make cache vector in class and add if(cache.find(s)!=cache.end) return cache[cache.find()]; before line 14
On line 13, add if(cache.find(s) != cache.end()) return cache[s]; After line 18, add cache[s] = rv; On line 33, add cache[s] = rv before return On line 37 add cache[s] = rv * get() before return.
Use depth first search
x


cache
line
add
return
long
vector
string
map
rv
value
37
check
Create
Store
call
returning
recursive
function
recursion
key
class
cacheend
cachefinds
found
already
ifcachefinds
cacherv
insert
caches
something
declare
search
passed
mapstring
cachekey
cacheszero
like
Idk
cachestring
first
int
save
statement
answer
see
find
calculated
19
33
31
12
13
14
4
3
1
10
values
Initialize
size
push
depth
result