**Quiz - Paper 5**

**Instructions**

**- This quiz contains 25 multiple-choice questions.**

**- Select the best answer for each question.**

**- Time allowed: 30 minutes.**

bleedecode bleedecode bleedecode bleedecode bleedecode bleedecode bleedecode bleedecode bleedecode bleedecode bleedecode bleedecode bleedecode bleedecode bleedecode bleedecode bleedecode bleedecode bleedecode bleedecode bleedecode bleedecode bleedecode bleedecode bleedecode bleedecode bleedecode bleedecode

1. **What will printf("%d", 2 + 1 == 3); output in C?**
   * A) 1
   * B) Segmentation fault (core dumped)
   * C) "21" because C is weird
   * D) "True" because C supports booleans (lol, no)
2. **What does int \*p = NULL; printf("%d", \*p); do?**
   * A) Prints 0, obviously
   * B) Crashes harder than my hopes of debugging in one try
   * C) Calls my professor and screams for help
   * D) Prints some random garbage value because C loves chaos
3. **What does this C snippet do?**

char \*ptr = "Hello";

ptr[0] = 'M';

* + A) Changes "Hello" to "Mello"
  + B) Causes a segmentation fault
  + C) Calls printf("Why did you do this?") automatically
  + D) Summons undefined behavior demons

1. **In C, what’s the real difference between malloc() and calloc()?**
   * A) malloc() gives you whatever is in memory, calloc() is nice and zeroes it out
   * B) calloc() is just malloc() but fancier
   * C) One is for smart programmers, one is for reckless ones
   * D) malloc() gives you memory, calloc() gives you existential dread
2. **What does printf("%d", sizeof('\0')); output?**
   * A) 1
   * B) 4 (because it’s treated as an int)
   * C) "You really thought that would be easy?"
   * D) Segmentation fault (core dumped)
3. **What does "this code slaps" mean?**
   * A) It's really good
   * B) It physically assaulted me during debugging
   * C) It contains so many hacks it should be illegal
   * D) It only works when my boss isn’t looking
4. **When someone says your solution is "big brain," what do they mean?**
   * A) It’s brilliant but unnecessarily complex
   * B) It looks smart but actually doesn't work
   * C) It runs, but nobody knows how or why
   * D) You just reinvented printf() from scratch
5. **What does it mean to "yeet" code?**
   * A) Delete it aggressively
   * B) Deploy it without testing
   * C) Submit it for review and log off immediately
   * D) Rename main() to why() and move on
6. **What's a "Karen" in tech support?**
   * A) A user who thinks "turning it off and on again" is offensive
   * B) A compiler that refuses to accept your code
   * C) Someone who files a bug report saying "It’s broken" with no details
   * D) That one teammate who never merges their pull requests
7. **What does "gatekeeping" mean in programming communities?**

* A) Arguing that real programmers only use Vim
* B) Refusing to explain something because “it’s basic”
* C) Believing only 90s kids remember pointers
* D) All of the above, and yes, you’re guilty

1. **What's the half-life of free pizza at a college coding event?**

* A) Logarithmic decay based on the number of CS students
* B) Theoretical, because it vanishes instantly
* C) If it's pineapple pizza, infinite
* D) Negative, because seniors take slices before the event starts

1. **What's the correlation between approaching deadlines and Stack Overflow visits?**

* A) Exponentially increasing panic
* B) Visits remain constant; tabs increase
* C) Strong positive correlation, peaking at 3 AM
* D) You just copy-paste from ChatGPT now

1. **What's the universal truth about group projects?**

* A) One person does everything
* B) Git logs reveal the real contributor (spoiler: it’s not the guy talking the most)
* C) The deadline is the only thing keeping the team together
* D) All of the above, and we all know who the slacker is

1. **When does a CS student's impostor syndrome peak?**

* A) First internship, when they pretend to understand meetings
* B) When debugging takes longer than writing the code
* C) When a junior fixes their bug in 2 minutes
* D) It never stops, just like memory leaks

1. **What’s the best way to tell if someone’s a good programmer?**

* A) How many times they say “this should be working”
* B) Their caffeine intake levels
* C) How often they reinstall Linux for fun
* D) Their ability to write readable C code (trick question: no one does)

1. **If you write perfect code but no one reviews it, does it exist?**

* A) No, unreviewed code is always broken
* B) Yes, but you’ll break it trying to explain it
* C) Yes, but good luck proving it
* D) This is theoretical—perfect code doesn’t exist

1. **What happens when you document your code too well?**

* A) The code changes, but the documentation never does
* B) You get promoted to a non-coding role
* C) People suspect you of hiding something
* D) Future devs still won’t read it

1. **How many developers does it take to choose a C standard?**

* A) Just one, but they'll argue between C89 and C99 forever
* B) An infinite loop of debates
* C) Just use gcc -std=whatever and pray
* D) No one chooses, they just blame undefined behavior

1. **If a program works but no one understands why, what's the best strategy?**

* A) Slap a "DO NOT TOUCH" comment on it
* B) Hope it never needs updates
* C) Call it "legacy code" and pretend it's someone else’s problem
* D) All of the above, depending on how much you care

1. **What's the relationship between how critical a piece of C code is and how ugly it is?**

* A) The more important, the uglier
* B) The cleaner it looks, the more it’s hiding
* C) If it’s critical *and* pretty, you probably copied it from somewhere
* D) No correlation, everything in C looks scary

1. **What's the most accurate way to measure a C programmer's experience?**

* A) The number of hours they’ve spent debugging pointers
* B) Their ability to write printf() without Googling
* C) The depth of their hatred for malloc() bugs
* D) Their emotional reaction to the word "segmentation fault"

1. **What happens when you try to fix one small C bug?**

* A) You introduce three new ones
* B) The program stops compiling
* C) The bug disappears mysteriously, only to return later
* D) All of the above, in order

1. **How do you know a C program is production-ready?**

* A) It compiles without warnings
* B) It hasn't crashed for at least 24 hours
* C) The original developer left the company
* D) Nobody understands how it works, so it must be done

1. **What happens when you explain your bug to a colleague?**

* A) You figure it out mid-sentence
* B) They spot the problem immediately, like a wizard
* C) The code starts working for no reason
* D) All of the above, and you hate it

1. **What's the fate of all "temporary workarounds"?**

* A) They become permanent
* B) They cause more problems than they solve
* C) Someone calls them an "architectural decision"
* D) All of the above, and management approves it