CS-392 Midterm Review!

Testing Point #1: **C programming basics**: *types *arrays + pointers *headers

Types: int ⇒ %d, char ⇒ %c, dataType arrayName[arraySize];

```
//input & store in 3rd loc        // print first element
scanf("%d", &mark[2])             printf("%d", mark[0])
```

Arrays + Pointers: if we have a variable var, &var gives the add.

[printf("var: %d\n", var) vs. printf("var: %p", &var)]

Output: var: 5 ← var : 2686778 →

Pointer Syntax: int* p or int *p or int * p

↳ store addresses rather than values

Pointers Examples: int* pc, c;

c = 22 ⇒ &c = address of c, c = value of c

pc = &c ⇒ pc = address pointerpc, *pc = content of pnt

c = 11 ⇒ pc = " " *pc = content (11)

*pc = 2 ⇒ &c = address of c, c = value of c, (2)

Determine the role of * and & through examples ↻

Arrays + Pointer pt2: &x[i] is equal to x+i, x[i] is equal *(x+1)

int x[5] = {1,2,3,4,5}; int* ptr; ptr = &x[2]

When we do *ptr ⇒ 3 is the returned val

Header Files:

```
#include thefilename.c
void function1(char *input, char *out);
int  function2(const void *x);
```

Testing Point #2: **C Programming Strings**: *What is a string? *Common APIs of strings

String Basics: char c[] = "c string";

c		s	t	r	i	n	g	\0

declare → char s[5]; initialize → char c[] = "abcd"

scanf → read till white ↑ length +1? Because we have a \0

fgets → line of string, ↘ will only get 1st word, puts → display

Commonly Used Functions:

- strlen() → calc length of str, takes string → longint
- strcopy() → copy to other array, takes char* dest, source
- strcmp() → ret 0 if both str are identical, 2 char*
- strcat() → concats 2 strs, stores joint in 1st char*
- memcopy() → (dest, *src, n) = n characters to mem area src to memory area dest

Keep in mind all these take various param!

Testing Point #3: Standard I/O & Stream I/O: *Basics * Open, Read, Write *Size

File Operations: 1. Create New File
2. Open Existing File
3. Closing a File
4. Reading / Writing to a File
] FILE *fptr

Creation/Edit Files: * fopen() → ("file", mode) mode= r-read, w-write, r+ = both
* fclose(ptr) → close w/ associated pointer

Reading & Writing to File: * we use fprintf() and fscanf()
↳ file version of printf & scanf, we want pointer
* fread() & fwrite() for binary files
↳ fwrite(addressData, sizeData, numsberData, ptr)
* fseek() → seeks the cursor to given file
↳ fseek(FILE *stream, long int offset, int whence)
ptr size SEEK_END
*rewind(FILE*stream) → position indicator to start

Testing Point #4: Dynamic Memory Allocation: * How to malloc and free memory

Basics: Once the size of an array is declared → can't change (duh!)
- if it's insufficient → allocate memory during runtime

Library Functions: * malloc(size) ⇒ reserves block of mem in bytes, return ptr
Ex: ptr = (float) malloc (100 * sizeof(float));
400 bytes stored = 100 * 4 bytes
* free(ptr) ⇒ frees the allocated memory pointed

Testing Point #5: Error Handling: * Check if fopen actually works

Example: FILE *fp; fp = fopen("file.txt", "rb");
if(fp == NULL){ exit(1) }
→ Also exit(EXIT_FAILURE), erno for specific