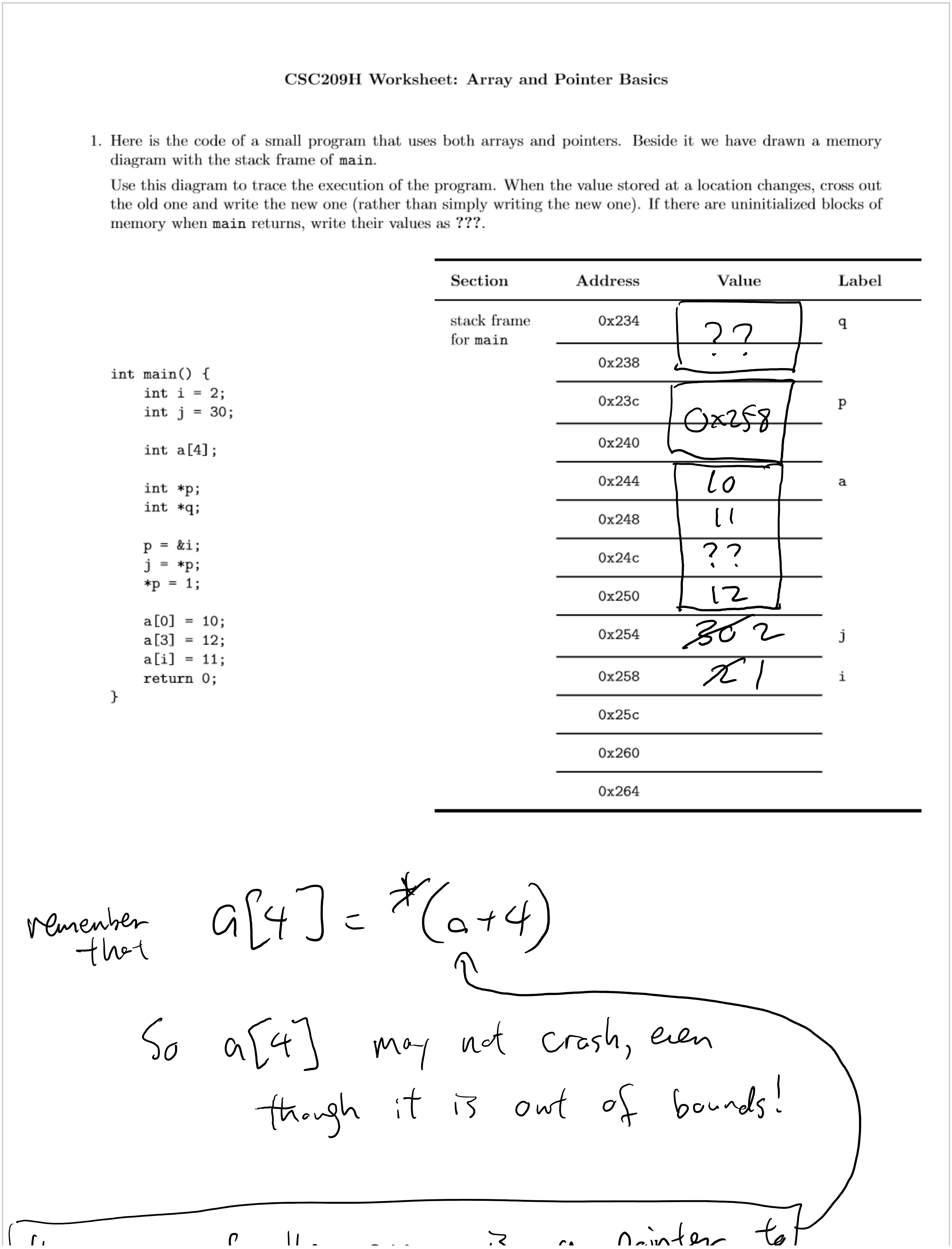
2.1

January 13, 2025



CSC209H Worksheet: Array and Pointer Basics

1. Here is the code of a small program that uses both arrays and pointers. Beside it we have drawn a memory diagram with the stack frame of `main`.

   Use this diagram to trace the execution of the program. When the value stored at a location changes, cross out the old one and write the new one (rather than simply writing the new one). If there are uninitialized blocks of memory when `main` returns, write their values as ???.

```
int main() {
    int i = 2;
    int j = 30;

    int a[4];

    int *p;
    int *q;

    p = &i;
    j = *p;
    *p = 1;

    a[0] = 10;
    a[3] = 12;
    a[i] = 11;
    return 0;
}
```

Section	Address	Value	Label
stack frame for `main`	0x234	??	q
	0x238		
	0x23c	0x258	p
	0x240		
	0x244	10	a
	0x248	11	
	0x24c	??	
	0x250	12	
	0x254	~~30~~ 2	j
	0x258	~~2~~ 1	i
	0x25c		
	0x260		
	0x264		

remember that $a[4] = *(a+4)$

So $a[4]$ may not crash, even though it is out of bounds!

[illegible] pointer to

the name of the array is a pointer to its first element !

CSC209H Worksheet: Array and Pointer Basics

2. Each example below contains an independent code fragment. In each case there are variables **x** and **y** that are missing declaration statements. In the boxes to the right of the code write declaration statements so that the code fragment would compile and run without warnings or errors.

Code Fragment	Declaration for x	Declaration for y

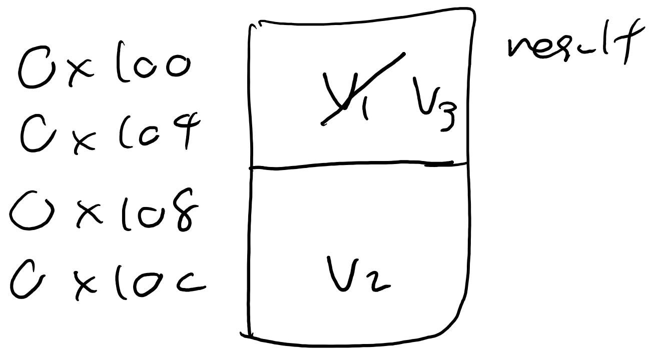
`x = 10;`
`y = 'A';`
	int x;	char y;

`int age = 99;`
`x = &age;`
`y = *x;`
	int *x;	int y;

`double *p;`
`x = &p;`
`y = &x;`
	double **x;	double ***y;

`float f = 4.5;`
`float *p = &f;`
`x = &p;`
`y = **x;`
	float **x;	float y;

↙ array of char*
`char *result[2];`
`x = result[0];`
`// some hidden code`
`result[0] = "read only";`
` y = x[0];`
	char * x;	char y;



assuming v_1, v_2 are some arbitrary values.

v_3 is a pointer to the string (more on this when we talk about strings)

$\boxed{x = v_1}$

$$x[0] = *(x+0) = *x$$