Tim Tanasse
CSCD 240
May 13, 2015
Notes # #

```
int **pptr = grades;
printf("-1: pptr= %p\n", pptr);
printf("-1: &pptr[0] = %p\n", &pptr[0] );
printf("-1: pptr+1= %p\n", pptr + 1);
printf("-1: pptr+2= %p\n", pptr + 2);
```

Explanation: the contents of **pptr** and the result of &pptr[0] are the same, because pptr contains the address of **pptr[0]**: 0x7FB322403930. **pptr** points to integer pointers, which have a size of 8. Therefore the address of **pptr + 1** has an address of 0x7FB322403938, and **pptr + 2** has an address of 0x7FB322403940.

```
printf("0: pptr[0]= %p\n", pptr[0]);
printf("0: *pptr= %p\n", *pptr);
printf("0: &pptr[0][0]= %p\n", &pptr[0][0]);
```

Explanation: The contents of pptr[0] is the same as **grades[0]** and has an address of 0x7FB3224000E0. the dereferenced **pptr** and pptr[0] have the same value, which was recorded previously. The same is true of the next expression **pptr[0][0]** which is equivalent of **&(*(*(pptr + 0) + 0))**. This translates similarly for **pptr[0]**, which is the same as ***(pptr + 0)** which is the same as ***pptr**

```
printf("1: pptr[1]= %p\n", pptr[1]);
printf("1: *(pptr + 1)= %p\n", *(pptr + 1));
printf("1: &pptr[1][0] = %p\n", &pptr[1][0] );
```

Explanation: The contents of **pptr[1]** is a memory address that is not contiguous of the previous addresses, that is, the contents of **pptr[1]** cannot be known by knowing the contents of *pptr, unlike the rest of pointer arithmetic, where any address can be known by knowing the base address. This is because every position of pptr (**pptr[0]**, **pptr[1]** ...) is allocated individually. These pointers do however point to contiguously allocated memeory. The three memory addresses are the same, for the same reasons as all of 0 :, they are equivalent statements: 0x7FB322401F50.

```
printf("2: *pptr + 1 = %p\n", *pptr + 1);
printf("2: *(pptr+0) + 1 = %p\n", *(pptr + 0) + 1);
printf("2: &pptr[0][1] = %p\n", &pptr[0][1] );
```

Explanation: As stated in 0, ***(pptr + 0)** is equivalent to ***pptr**, thus the addresses for ***(pptr + 0) + 1** and ***pptr + 1** are the same: 0x7FB3224000E4. This is because *pptr is an integer pointer, and not a pointer pointer, and integers have a size of 4 bytes. Similar to before **pptr[0][1]** is equivalent to ***(*(pptr + 0) + 1)**, and &*(*(pptr + 0) + 1) is the same as ***(pptr + 0) + 1**, and holds the same address as mentioned previously: 0x7FB3224000E4.

```
printf("3: *(pptr[1] + 1) = %d\n", *(pptr[1] + 1) );
printf("3: *( *(pptr + 1) + 1) = %d\n", *( * (pptr + 1) + 1) );
printf("3: pptr[1][1] = %d\n", pptr[1][1] );
```

Explanation: These statements are all equivalent forms of **pptr[1][1]** which is the integer contained at the memeory location ***(pptr + 1) + 1**, which is 5. This does not need much more explanation, since these equivalences have been shown multiple times above.