Extra Exercise

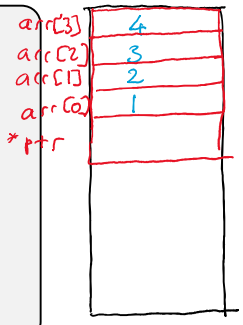
- Use a box-and-arrow diagram for the following program and explain what it prints out:

```c
#include <stdio.h>

int foo(int* bar, int** baz) {
  *bar = 5;
  *(bar+1) = 6;
  *baz = bar + 2;
  return *((*baz)+1);
}

int main(int argc, char** argv) {
  int arr[4] = {1, 2, 3, 4};
  int* ptr;

  arr[0] = foo(&arr[0], &ptr);
  printf("%d %d %d %d %d\n",
         arr[0], arr[1], arr[2], arr[3], *ptr);
  return 0;
}
```



First we draw the stack frame for main

Extra Exercise

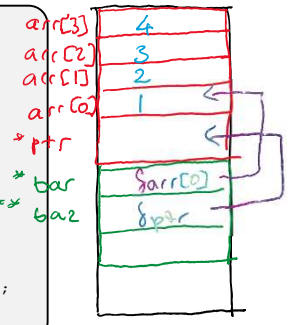
- Use a box-and-arrow diagram for the following program and explain what it prints out:

```
#include <stdio.h>

int foo(int* bar, int** baz) {
  *bar = 5;
  *(bar+1) = 6;
  *baz = bar + 2;
  return *((*baz)+1);
}

int main(int argc, char** argv) {
  int arr[4] = {1, 2, 3, 4};
  int* ptr;

  arr[0] = foo(&arr[0], &ptr);
  printf("%d %d %d %d %d\n",
         arr[0], arr[1], arr[2], arr[3], *ptr);
  return 0;
}
```



When foo function is executed, address of arr[0] will be passed to bar and address of ptr will be passed to baz

Extra Exercise

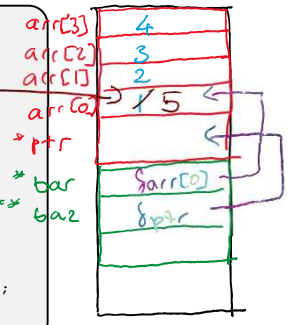
- Use a box-and-arrow diagram for the following program and explain what it prints out:

```
#include <stdio.h>

int foo(int* bar, int** baz) {
  *bar = 5;
  *(bar+1) = 6;
  *baz = bar + 2;
  return *((*baz)+1);
}

int main(int argc, char** argv) {
  int arr[4] = {1, 2, 3, 4};
  int* ptr;

  arr[0] = foo(&arr[0], &ptr);
  printf("%d %d %d %d %d\n",
         arr[0], arr[1], arr[2], arr[3], *ptr);
  return 0;
}
```



Dereferenced bar, which means arr[0] is changed to 5

Extra Exercise

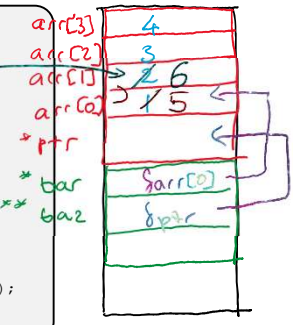
- Use a box-and-arrow diagram for the following program and explain what it prints out:

```
#include <stdio.h>

int foo(int* bar, int** baz) {
  *bar = 5;
  *(bar+1) = 6;
  *baz = bar + 2;
  return *((*baz)+1);
}

int main(int argc, char** argv) {
  int arr[4] = {1, 2, 3, 4};
  int* ptr;

  arr[0] = foo(&arr[0], &ptr);
  printf("%d %d %d %d %d\n",
         arr[0], arr[1], arr[2], arr[3], *ptr);
  return 0;
}
```



bar +1, is the address of arr[1]
Dereferenced(bar+1) which means arr[1] is changed to 6

Extra Exercise

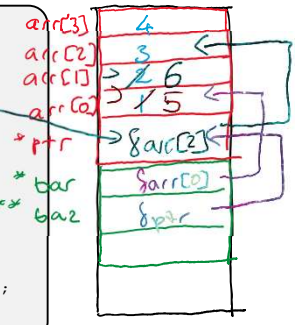
- Use a box-and-arrow diagram for the following program and explain what it prints out:

```
#include <stdio.h>

int foo(int* bar, int** baz) {
  *bar = 5;
  *(bar+1) = 6;
  *baz = bar + 2;
  return *((*baz)+1);
}

int main(int argc, char** argv) {
  int arr[4] = {1, 2, 3, 4};
  int* ptr;

  arr[0] = foo(&arr[0], &ptr);
  printf("%d %d %d %d %d\n",
         arr[0], arr[1], arr[2], arr[3], *ptr);
  return 0;
}
```



Dereferenced baz is ptr value, initially ptr value was garbage
We now change it to bar +2, which is the address of arr[2]

Extra Exercise

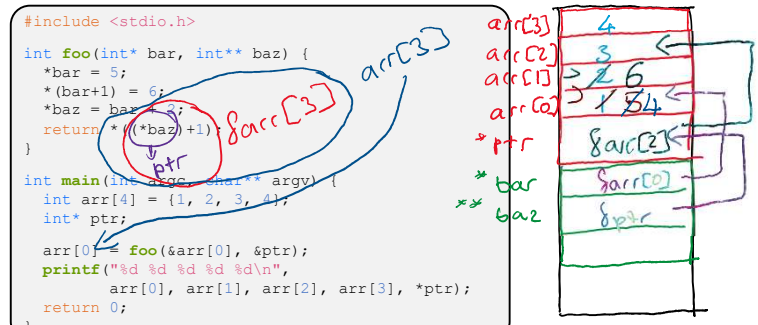
- Use a box-and-arrow diagram for the following program and explain what it prints out:

```
#include <stdio.h>

int foo(int* bar, int** baz) {
  *bar = 5;
  *(bar+1) = 6;
  *baz = bar + 3;
  return *((*baz)+1);
}

int main(int argc, char** argv) {
  int arr[4] = {1, 2, 3, 4};
  int* ptr;

  arr[0] = foo(&arr[0], &ptr);
  printf("%d %d %d %d %d\n",
         arr[0], arr[1], arr[2], arr[3], *ptr);
  return 0;
}
```



Dereferenced baz is ptr value
Ptr value + 1 is the address of arr[3]
Dereferenced address of arr[3] is *&arr[3] which is same as arr[3] so foo function returns arr[3] into arr[0]
So arr[0] value is changed to 4

Extra Exercise

- Use a box-and-arrow diagram for the following program and explain what it prints out:

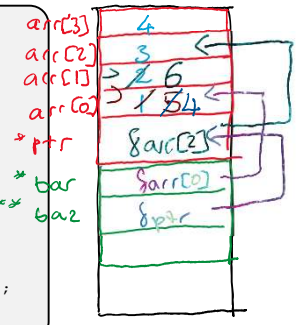
```c
#include <stdio.h>

int foo(int* bar, int** baz) {
  *bar = 5;
  *(bar+1) = 6;
  *baz = bar + 2;
  return *((*baz)+1);
}

int main(int argc, char** argv) {
  int arr[4] = {1, 2, 3, 4};
  int* ptr;

  arr[0] = foo(&arr[0], &ptr);
  printf("%d %d %d %d %d\n",
         arr[0], arr[1], arr[2], arr[3], *ptr);
  return 0;
}
```

4 6 3 4 3



Then print function prints arr[0] through arr[3] and arr[2]