**CS224 – Parameter Passing**

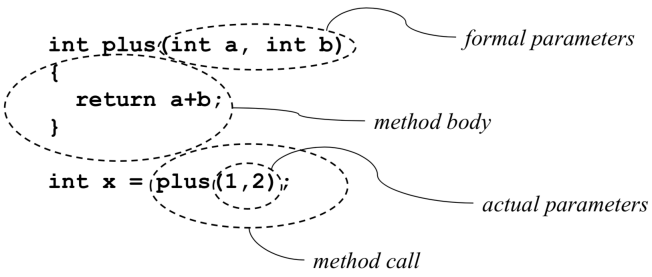
**Purpose:** There are several different mechanisms that programming language use for parameter passing from the call to the subprogram. We will examine these mechanisms and how different mechanisms may change the outcome.

**Knowledge:** This activity will help you become familiar with the following content knowledge:

- Parameter passing mechanisms

**Activity:** With your group perform the following tasks and answer the questions. You will be reporting your answers back to the class in 60 minutes.

1. The following diagram illustrates the components involved in parameter passing. How does the language determine which formal parameters go with which actual parameters?

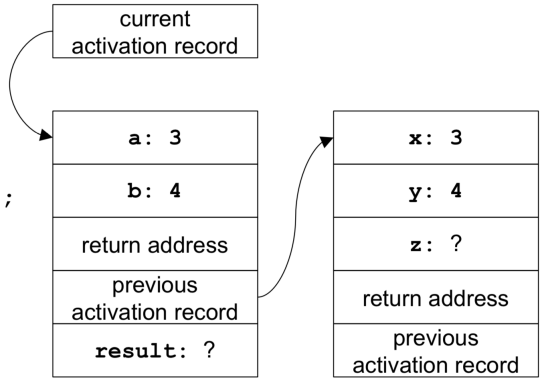


2. The mechanism of "Pass by Value" copies the value of the actuals to the formals within the activation record. Consider the following:

```
int plus(int a, int b) {
   a += b;
   return a;
 }

void f() {
  int x = 3;
  int y = 4;
  int z = plus(x, y);
}
```

What are the values of `x`, `y`, and `z` after the call of `plus` inside of `f`?




;

3. The language C uses pass by value. Take a look at the following C example and determine the values of `a` and `b` after `swap` is called. Be careful as it may not behave as you expect!

```
void swap(int x, int y) {
  int z;
  z=x; x=y; y=z;
}
void f() {
  int a = 3;
  int b = 4;
  swap(a, b);
}
```

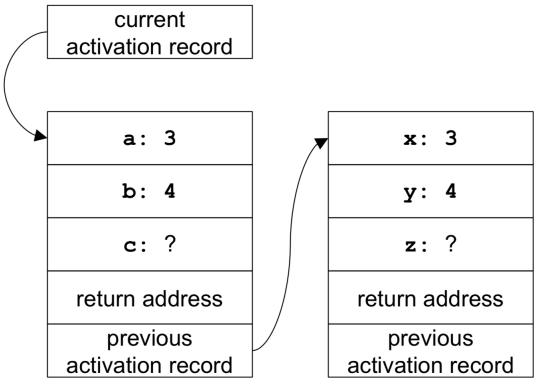
4. To fix the behavior of the above example, the language C has two operators `*` and `&`. The expression `&x` returns the address in RAM of the variable `x`. The expression `*y` returns the value contained at the address stored in variable y.

   Determine the values of `a` and `b` after `swap` is called with this new version.

```
void swap(int* px, int* py) {
  int z;
  z=*px; *px=*py; *py=z;
}
void f() {
  int a = 3;
  int b = 4;
  swap(&a, &b);
}
```

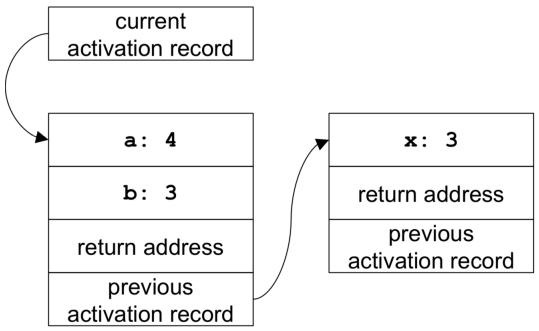
5. The mechanism "Pass by Result" does copying in the opposite direction. After a subprogram is done executing, the value of the formal is copied into the value of the actual. Consider this example where the parameters `a` and `b` are pass by value and the parameter `c` is pass by result. What are the values of `x`, `y` and `z` after the function `plus` is called?

```
void plus(int a, int b, by-result int c) {
  c = a+b;
}
void f() {
  int x = 3;
  int y = 4;
  int z;
  plus(x, y, z);
}
```



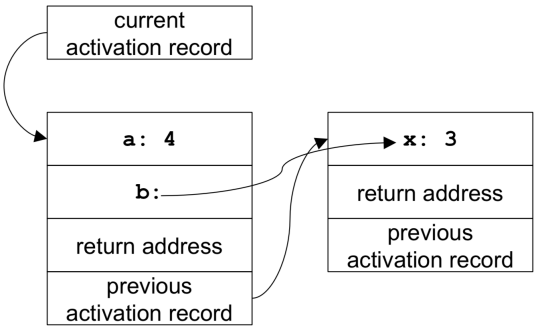
6. Combining both pass by value and pass by result we get the mechanism "Pass by Value-Result". In this mechanism, the actuals are copied to the formals, the subprogram is executed, and finally the formals are copied back to the actuals. Consider the following and determine the value of `x` after the execution of `plus`.

```
void plus(int a, by-value-result int b) {
  b += a;
}
void f() {
  int x = 3;
  plus(4, x);
}
```



7. Another mechanism is "Pass by Reference" in which the formals are pointers to the actuals. This means that when a formal parameter is modified in the subprogram, the actual parameter is the thing that get modified. Consider the following and determine the value of `x` after the execution of `plus`.

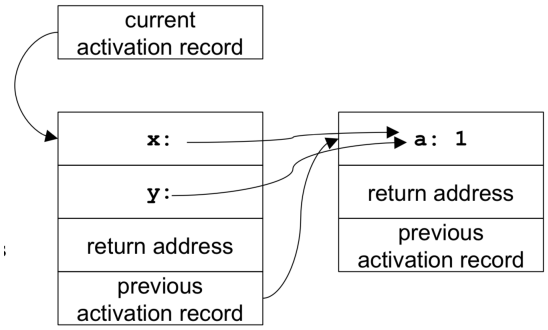
```
void plus(int a, by-reference int b) {
  b += a;
}
void f() {
  int x = 3;
  plus(4, x);
}
```



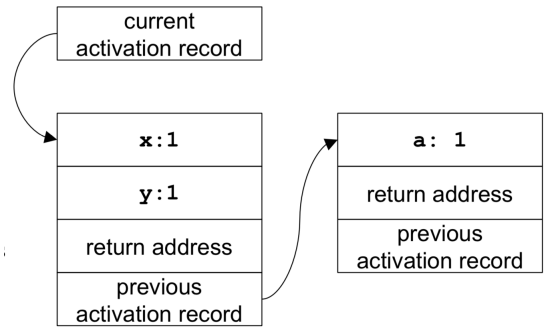
8. On the surface it may seem that the mechanisms of pass by value-result and pass by reference achieve the same thing so we will compare them closely. Consider the following example giving the value of `a` after the call of `p`. Do this first by pass by reference as shown in the first diagram, and then do it for pass by value-result as shown in the second diagram. You should see that the outcomes are not the same!

```
void p (int x, int y) {
   x++; y++;
}
void f() {
  int a = 1;
  p(a, a);
```

Pass by Reference:




Pass by Value-Result



9. While not true parameter passing, the next mechanism of "Macro Expansion" is an important stepping stone. In macro expansion, the body of the macro is evaluated in the caller's context. Each actual parameter is evaluated on every use of the corresponding formal parameter.

   For example in C we can define a macro MIN and then use it. The text of the macro gets expanded and then that text is executed:

```
#define MIN(X,Y) ((X)<(Y)?(X):(Y))
  a = MIN(b,c);
```

   What gets expanded is `a = ((b)<(c)?(b):(c))`

   What do you think gets expanded with:

```
  a = MIN(b++,c++)
```

   One of the big problems with macro expansion is something called "capture". Since the macro is just doing text substitution, it is possible that some of the text comes in conflict with identifier names in the caller. For example,

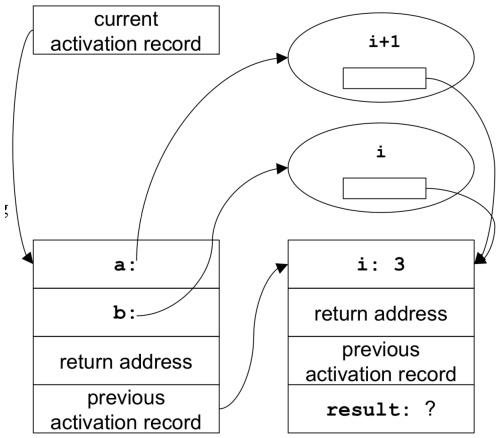
```
#define intswap(X,Y) {int temp=X; X=Y; Y=temp;}
int main(){
  int temp=1, b=2;
  intswap(temp,b);
  print("%d, %d\n", temp,b);
}
```

   Perform the macro expansion on this example and see if it does what the programmer intended, which is to swap `temp` and `b`.

10. The mechanism of "Pass by Name" is expansion without the capture. Consider the following example. What actually gets passed is a "thunk" which is a function containing the text of the actual parameter.

```
void f(by-name int a, by-name int b) {
  b=5;
  b=a;
}

int g() {
  int i = 3;
  f(i+1,i);
  return i;
}
```



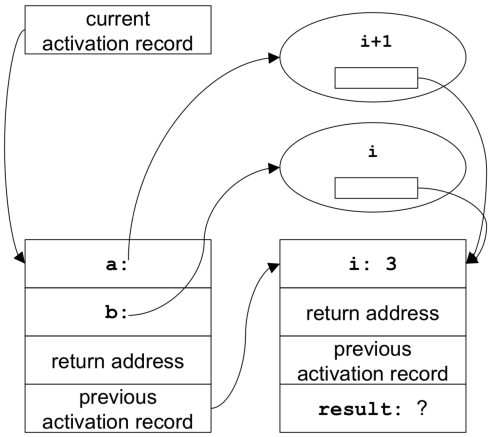

Verify that g will return the value of 6.

11. Use pass by name in this example and determine what will get printed. This example uses a global variable `i` which is accessed by both functions.

```
void f(x){
       i++;  // global reference
       x++;
}
void g(){
       i=1;  // global reference
       int [] a = new int[3];
       a[1]=1; a[2]=2;
       f(a[i]);
       print(a[1],a[2]);
}
```

12. The final mechanism is "Pass by Need". This mechanism is like pass by name but the evaluated formula is cached so it is not evaluated more than once. Consider the following example:

```
void f(by-need int a, by-need int b) {
  b=a;
  b=a;
}
void g() {
  int i = 3;
  f(i+1,i);
  return i;
}
```




If we were using pass by name `g` would return 5 since `i+1` is evaluated twice for the two assignments `b=a`. In constrast, pass by need would only evaluated `a` once and use that cached result in the second assignment. What would be returned for pass by need?