CPSC 213
Answers to Part II of Midterm Practice Questions

Part II

Question 1

```
ld  $b, r0          # r0 = address of b

ld 0x0(r0), r0      # r0 = address of array b

ld 0x8(r0), r1      # r1 = *(b+2)

ld $i, r2           # r2 = address of i

ld 0x0(r2), r2      # r2 = i

ld $a, r3           # r3 = address of a

st r1, (r3, r2, 4)  # a[i] = r1
```

Question 2

a. We assume:

- r5 points to the location for j and k is in the next word after j
  That is, we assume that foo stored r6 on the stack first and stored its local variables after that, so r5 points to the first local variable.

```
ld  $0x0 r0         # r0 = 0

ld  $i, r1          # r1 = address of i

st r0, 0x0(r1)      # i = 0

st r0, 0x0(r5)      # j = 0

st r0, 0x4(r5)      # k = 0
```

b. The complete code for foo may look like the following:

```
dec r5              # create stack space for saving r6
st r6, 0x0(r5)      # store r6 at the stack
deca r5             # create sack space for k
deca r5             # create sack space for j
st $bar, r0         # r0 = address of bar
gpc r6
inca r6             # r6 has the return address
j 0x0(r0)           # goto bar
ld  $0x0 r0         # r0 = 0
ld  $i, r1          # r1 = address of i
st r0, 0x0(r1)      # i = 0
st r0, 0x0(r5)      # j = 0
st r0, 0x4(r5)      # k = 0
inca r5             # relese stack space for j
inca r5             # relese stack space for k
ld 0x0(r5), r6      # restore r6 from the stack
inca r5             # relese stack space for r6
j 0x0(r6)           # return
```

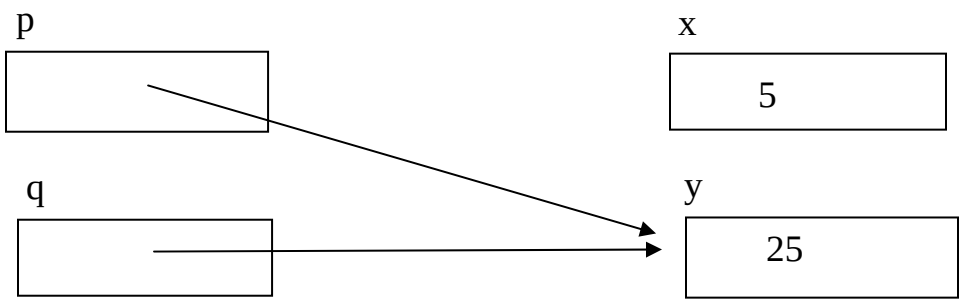
Question 3

Note: foo does not need to store r6 on the stack, as it does not call any other function

```
ld  $0x0 r0          # r0 = 0

ld  $i, r1           # r1 = address of i

st r0, 0x0(r1)       # i = 0

st r0, 0x8(r5)       # m = 0 , m is stored just below k

st r0, 0x0(r5)       # j = 0

st r0, 0x4(r5)       # k = 0
```

Question 4

The following diagram shows the final values:



Question 5

The C code that correspond to the given assembly code is:

```
int sum (int* a, int aLength) {
     int i;
     int s;
     s = 0;
     for (i=0; i<aLength; i++)
          s += a[i];
     return s;
}
```

Question 6

```
typedef struct {
     int  noSides;          //number of sides
     float *   side;        // a dynamic array with the length of each side
} Polygon;


Polygon* createPolygon( int n ) {
     Polygon* p = (Polygon *) malloc( sizeof(Polygon));
     p->noSides = n;
     p->side = (float *) malloc( n * sizeof(float));
     return p;
}


setSide(Polygon* pg, int s, float length) {
     if ( 0 <= s && s < pg->noSides )
          pg->side[s] = length;
}

     . . .
```