

(5 pts) Exercise 3-21

- Convert the following C code to MIPS:

```
float pick (float G[], int index) {  
    return G[index];  
}
```

(5 pts) Exercise 3-22

- Convert the following C code to MIPS:

```
float max (float A, float B) {  
    if (A > B) return A / B;  
    else         return B / A;  
}
```

(5 pts) Exercise 3-23

- Convert the following C code to MIPS:

```
float sum (float A[], int N) {  
    int j;  
    float sum = 0.0;  
    for (j=0; j<N; j++)  
        sum = sum + A[j];  
    return sum;  
}
```

(10 pts) Exercise 3-25

- Convert the following C code into MIPS.

```
float function2 (float x, float y) {  
    if (x > y)  
        return x + y;  
    else  
        return x - y;  
}
```

(20 pts) Exercise 3-26

- Convert the following C code into MIPS. A C float is stored as a MIPS single precision floating point value.

```
float dotproduct (float A[], float B[]) {  
    float sum = A[0] * B[0];  
    int ii;  
    for (ii = 1; ii < 20; ii++) {  
        sum = sum + A[ii] * B[ii];  
    }  
    return sum;  
}
```

(10 pts) Exercise 3-27

- Convert the following C code into MIPS. ASSUME that the result of multiplying g by h will always fit in just 32 bits.

NOTE: using **integers**, not floats, here!

```
int function6 (int g, int h) {  
    int prod = g * h;  
    if (prod < 0)  
        prod *= -1;  
    return prod;  
}
```

(3 pts EXTRA CREDIT) Exercise 3-31

- Convert the following C code to MIPS:

```
float average (float A[], int N) {  
    int j;  
    float sum = 0.0;  
    for (j=0; j<N; j++)  
        sum = sum + A[j];  
    return sum / N;  
}
```