Homework 1

Problem 1

Consider the following functions:

```
int min(int x, int y ){return x < y ? x : y;}
int max(int x, int y ){return x >y ? x : y;}
void decre(int *p, int v){*p -= v;}
int square(int x){return x*x;}
```

The following three code fragments call these functions:

```
A. int i= min(x, y );
while(i< max(x, y))
{
t += square(i);
decre(&i,-1);
}
B. int i= max(x, y);
while(i>= 0 )
{
t += square(i);
decre(&i,2);
}
C. for(i=max(x, y ); i> min(x, y); decre(&i,1))
t+= square(i);
```

Assume x equals 1 and y equals 10. Fill in the following table indicating the number of times each of the four functions is called in code fragment A-C:

Code	Min	max	decre	Square
A				
B				
C				

Problem 2

In this chapter we will take a single function and generate many different variants that preserve the function's behavior, but with different performance characteristics. For three of these variants, we found that the run times(in clock cycles) can be approximated by the following functions:

Version 1 60+30n
Version 2 135+4n
Version 3 150+2n

For what values of n would each version be the fastest of the three? **Remember that n will always be an integer.**

Problem 3

The following problem illustrates the way memory aliasing can cause unexpected program behavior. Consider the following procedure to swap two values:

```
1   /* move value x at xp to y at yp */
2   int move(int *xp, int *yp)
3   {
4       *yp = *xp; /* copy */
5       free(xp);
6       return *yp;
7   }
```

If this procedure is called with xp equal to yp, what effect will it have

Problem 4

Rewrite the code for prefixSum so that it does not need to repeatedly retrieve the value of p[i] from memory.

```
/*compute prefix sum of vector b*/
void prefixSum(float b[], float p[], long int n)
{
long int i;
p[0]= b[0];
for( i= 1; i< n; i++ )
p[i] = p[i-1] + b[i];
}
```