1 Problem 1

1.1 Part A

Code for hello world program

```
#include <stdio.h>
#include <math.h>

int main(int argc,char **argv){
printf("Hello World \n");
}
```

Output for hello world program

```
Hello World
```

1.2 Part B

Code for float formatting

```
#include <stdio.h>
#include <math.h>

int main(int argc,char **argv){
double nineteen_billion = 19.0*pow(10.0,9.0);
double phi = sqrt(5.0)/2.0 -.5;
printf("Nineteen billion  = %1.1e\n",nineteen_billion);
printf("The value of the golden mean is= %1.8e \n",phi);
}
```

The output for the float formatting program

```
Nineteen billion  = 1.9e+10
The value of the golden mean is= 6.18033989e-01
```

2 Problem 2

2.1 Part A

Explanation: To overflow an integer we will try to find a large power of two which cannot be represented. We know that an int is represented in so many bits and every sucessive doubling shifts a binary digit to the next place. Enough shifts will cause an overflow where there are no binary 'ones' in the binary array accept for an overflow binary digit in the sign place meaning that at an overflow the integer will take the value of zero, and that is what the code below will check for.

```
#include <stdio.h>
#include <math.h>
int int_power(int a, int b);
int main(int argc, char **argv){
    int j = 1;
    for(int i =1; i<=1000; i++){
       j=2*j;
       printf("2^%d = %d\n",i,j);
       if(j < 0){
          printf("Integer Overflow occured, now try numbers inbetween 2^%d and 2^%d\n",i
          int value_before = int_power(2,(i-2));  // go to value before upper limiit
          int k = value_before;
          int g = 0;
          while (k>0){
           k = k+1;
           g = g+1;
          }
          printf("The last availible integer is %d\n",g -1 +value_before);
          break;
       }
    }
}

int int_power(int a, int b){
int base =a;
for (int i =1; i<=b;i++){
 a=a*base;
}
return a;
}
```

This was the output for that code

```
2^1 = 2
2^2 = 4
2^3 = 8
2^4 = 16
2^5 = 32
2^6 = 64
2^7 = 128
2^8 = 256
```

```
2^9 = 512
2^10 = 1024
2^11 = 2048
2^12 = 4096
2^13 = 8192
2^14 = 16384
2^15 = 32768
2^16 = 65536
2^17 = 131072
2^18 = 262144
2^19 = 524288
2^20 = 1048576
2^21 = 2097152
2^22 = 4194304
2^23 = 8388608
2^24 = 16777216
2^25 = 33554432
2^26 = 67108864
2^27 = 134217728
2^28 = 268435456
2^29 = 536870912
2^30 = 1073741824
2^31 = -2147483648
Integer Overflow occured, now try numbers inbetween 2^30 and 2^31
The last availible integer is 2147483647
```

Therefore the last integer that can be represented is $2,147,483,647$ which is $2^31 - 1$. This means my compilier stores integers as 32 bit numbers.

2.2 Part B

Using the same method in part A, we try to overflow floats and doubles. We expect to see larger numbers as is the point with these types in the language. Code:

```
#include<stdio.h>
#include<math.h>

int main(int argc, char **argv){
    float j = 1.0;
    double g = 1.0;

    printf("Running float overflow test\n");
    //Loop through find value that causes float to have value infintiy
    for(int i =1; i<= 10000; i++){
       if(j != INFINITY){
       j=2*j;
       if(i<3 || i>124){
       printf("2^%d = %1.5e\n",i,j);
       }else if(i==3){
        printf("......\n");
       }
      }else{
       printf("Float Overflow happens at value 2^%d\n",i-1);
       break;
      }
    }

    printf("\n \nRunning double overflow test\n") ;
    //Loop through find value that causes double to have value infintiy
    for(int i =1 ; i<=10000;i++){
      if(g != INFINITY){
        g = 2*g;
         if(i<3 || i>1020){
           printf("2^%d = %1.5e\n",i,g);
       }else if(i==3){
           printf("......\n");
       }
      }else{
        printf("Double Overflow happens at value 2^%d\n",i-1);
        break;
      }
    }
}
```

Ouput:

```
Running float overflow test
2^1 = 2.00000e+00
2^2 = 4.00000e+00
......
2^125 = 4.25353e+37
2^126 = 8.50706e+37
2^127 = 1.70141e+38
2^128 = inf
Float Overflow happens at value 2^128


Running double overflow test
2^1 = 2.00000e+00
2^2 = 4.00000e+00
......
```

```
2^1021 = 2.24712e+307
2^1022 = 4.49423e+307
2^1023 = 8.98847e+307
2^1024 = inf
Double Overflow happens at value 2^1024
```

This indicates the largest float that can be represented with out overflow on my compilier is $2^{128} - 1$ which happens to be $3.4 * 10^{38}$. The largest double that can be represent with out overflow on my compilier is $2^{1024} - 1$ whcih happens to be $1.7 * 10^{308}$. This is because for the struture of floats is 32 bit, 7 are dedicated to the exponent, and 2^7 is 128. The double is a 64 bit number and 10 digits are provide to the exponent, and 2^10 is 1024.