Assignment 1

Solution1:

(a) Sign = 0;

Exponent = in binary (1110) = 8 + 4 + 2 + 0 = 14 (in decimal);

F = in binary (1010) = 2^-1 + 0 + 2^-3 = 0.625 (in decimal);

(b) Biased exponent = 14;

True exponent = 14-127 = -113;

(c) for 0<e<255; mantissa = 1+0.f;

Mantissa = 1.625;

(d) Decimal value = (-1) ^sign * 2^(Exponent-127) * 1.f;

= 2^ (-113) * 1.625;

= $1.56481808 \times 10^{-34}$;

Solution2:

(a)

rajeev_21111027_tut1.cpp

Codes > rajeev_21111027_tut1.cpp > main()

```cpp
int main()
{
    double num = -1;
    // underflow
    while (abs(num/2)>0)
    {
        num/=2;
    }
    cout<<"Lower limit for single-bit precision number "<<num<<endl;
    // overrflow
    num=1;
    while (isfinite(num*2)>0)
    {
        num*=2;
    }
    cout<<"Upper limit for single-bit precision number "<<num<<endl;


    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Code - Codes

```
rajeev-gupta@rajeevgupta-Inspiron-15-3565:~/Desktop/Fifth sem/PHN 323 NACP/Codes$ cd "/home/rajeev-gupta/Desktop/Fifth sem/PHN 323 NACP/Codes/" && g++ rajeev_21111027_tut1.cpp -o rajeev_21111027_tut1 && "/home/rajeev-gupta/Desktop/Fifth sem/PHN 323 NACP/Codes/"rajeev_21111027_tut1
Lower limit for single-bit precision number -4.94066e-324
Upper limit for single-bit precision number 8.98847e+307
rajeev-gupta@rajeevgupta-Inspiron-15-3565:~/Desktop/Fifth sem/PHN 323 NACP/Codes$
```

Lower limit for double-bit precision number -4.94066e-324

Upper limit for double-bit precision number 8.98847e+307

(b)

rajeev_21111027_tut1.cpp

Codes > rajeev_21111027_tut1.cpp > main()

```cpp
#include <bits/stdc++.h>
using namespace std;

int main()
{
    float num = -1;
    // underflow
    while (abs(num/2)>0)
    {
        num/=2;
    }
    cout<<"Lower limit for single-bit precision number "<<num<<endl;
    // overrflow
    num=1;
    while (isfinite(num*2)>0)
    {
        num*=2;
    }
    cout<<"Upper limit for single-bit precision number "<<num<<endl;

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Code - Codes

```
rajeev-gupta@rajeevgupta-Inspiron-15-3565:~/Desktop/Fifth sem/PHN 323 NACP/Codes$ cd "/home/rajeev-gupta/Desktop/Fifth sem/PHN 323 NACP/Codes/" && g++ rajeev_21111027_tut1.cpp -o rajeev_21111027_tut1 && "/home/rajeev-gupta/Desktop/Fifth sem/PHN 323 NACP/Codes/"rajeev_21111027_tut1
Lower limit for single-bit precision number -1.4013e-45
Upper limit for single-bit precision number 1.70141e+38
rajeev-gupta@rajeevgupta-Inspiron-15-3565:~/Desktop/Fifth sem/PHN 323 NACP/Codes$
```

Lower limit for single-bit precision number -1.4013e-45

Upper limit for single-bit precision number 1.70141e+38

(c)

```cpp
    int lower = -1;
    int upper = 1;
    // integers uses 2's complement, hence for a large values than the upperlimit of int
    // the computer will consider it as a negative number.
    while(upper>0){
        // cout<<upper<<endl;
        upper*=2;
    }
    cout<<"Upper limit for int number "<<upper-1<<endl;
    // similiarly the negative value of int will become positive
    //  as the value is decreased further
    while(1){
        int check = lower*2;
        if(check) lower=check;
        else break;
        // cout<<lower<<endl;
    }
    cout<<"Lower limit for int number "<<lower<<endl;

    return 0;
}
```

```
rajeev-gupta@rajeevgupta-Inspiron-15-3565:~/Desktop/Fifth sem/PHN 323 NACP/Codes$ cd "/home/rajeev-gupta/Desktop/Fifth sem/PHN 323 NACP/Codes/" && g++ rajeev_21111027_tut1.cpp -o rajeev_21111027_tut1 && "/home/rajeev-gupta/Desktop/Fifth sem/PHN 323 NACP/Codes/"rajeev_21111027_tut1
Upper limit for int number 2147483647
Lower limit for int number -2147483648
rajeev-gupta@rajeevgupta-Inspiron-15-3565:~/Desktop/Fifth sem/PHN 323 NACP/Codes$
```

Upper limit for int number 2147483647

Lower limit for int number –2147483648

Solution 3:

(a)

rajeev_21111027_tut1.cpp ● test.cpp Release Notes: 1.81.0

Codes > rajeev_21111027_tut1.cpp > main()

```cpp
    cout<<"Lower limit for int number "<<lower<<endl;
}
void solution3a(){
    double x = 1.0;
    double em = 1.0;

    while (1.0 + em != 1.0) {
        em = x;
        x /= 2.0;
    }
    cout << "Machine Precision of double-precision floats (εM_double): "
         << em << endl;
}
int main()
{
//    solution2a();
//    solution2b();
//    solution2c();
    solution3a();

    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Code - Codes

```
cd "/home/rajeev-gupta/Desktop/Fifth sem/PHN 323 NACP/Codes/" && g++ rajeev_21111027_tut1.cpp -o rajeev_21111027_tut1 &&
"/home/rajeev-gupta/Desktop/Fifth sem/PHN 323 NACP/Codes/"rajeev_21111027_tut1
rajeev-gupta@rajeevgupta-Inspiron-15-3565:~/Desktop/Fifth sem/PHN 323 NACP$ cd "/home/rajeev-gupta/Desktop/Fifth sem/PHN
323 NACP/Codes/" && g++ rajeev_21111027_tut1.cpp -o rajeev_21111027_tut1 && "/home/rajeev-gupta/Desktop/Fifth sem/PHN 323
 NACP/Codes/"rajeev_21111027_tut1
Machine Precision of double-precision floats (εM_double): 1.11022e-16
rajeev-gupta@rajeevgupta-Inspiron-15-3565:~/Desktop/Fifth sem/PHN 323 NACP/Codes$
```

Machine Precision of double-precision floats (εM_double): 1.11022e-16

(b)

```
rajeev_21111027_tut1.cpp   test.cpp   Release Notes: 1.81.0
Codes > rajeev_21111027_tut1.cpp > solution3b()
void solution3b(){
    complex<double> cmp = 1.0;
    complex<double> e_cmp = 1.0;

    while (1.0 + e_cmp != 1.0) {
        e_cmp = cmp;
        cmp /= 2.0;
    }

    cout << "Machine Precision of complex numbers (εM_complex): " << e_cmp << endl;

}

int main()
{
    // solution2a();
    // solution2b();
    // solution2c();
    // solution3a();
    solution3b();

    return 0;
}
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL                    Code - Codes
rajeev-gupta@rajeevgupta-Inspiron-15-3565:~/Desktop/Fifth sem/PHN 323 NACP/Codes$ cd "/home/rajeev-gupta/Desktop/Fifth se
m/PHN 323 NACP/Codes/" && g++ rajeev_21111027_tut1.cpp -o rajeev_21111027_tut1 && "/home/rajeev-gupta/Desktop/Fifth sem/P
HN 323 NACP/Codes/"rajeev_21111027_tut1
Machine Precision of complex numbers (εM_complex): (1.11022e-16,0)
rajeev-gupta@rajeevgupta-Inspiron-15-3565:~/Desktop/Fifth sem/PHN 323 NACP/Codes$
```

Machine Precision of complex numbers (εM_complex): (1.11022e-16,0)

Solution 4:

rajeev_21111027_tut1.cpp × machar.h Release Notes: 1.81.0

Codes > rajeev_21111027_tut1.cpp > solution4()

```cpp
    return ans;
}
void solution4(){
    int ibeta, it, irnd, ngrd, machep, negep, iexp, minexp, maxexp;
    float eps, epsneg, xmin, xmax;
    machar(&ibeta, &it, &irnd, &ngrd, &machep, &negep, &iexp, &minexp, &maxexp, &eps, &epsneg, &xr

    // Display the machine characteristics
    cout << "Machine Characteristics:" << endl;
    cout << "-----------------------------" << endl;
    cout << "Base (radix) of the machine (ibeta): " << ibeta << endl;
    cout << "Number of base-ibeta digits in the mantissa (it): " << it << endl;
    cout << "Rounding mode (irnd): " << irnd << endl;
    cout << "Guard digits (ngrd): " << ngrd << endl;
    cout << "Machine precision (machep): " << machep << endl;
    cout << "Negative of machine precision (negep): " << negep << endl;
    cout << "Number of bits in the exponent (iexp): " << iexp << endl;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Code - Codes

```
Machine Characteristics:
-----------------------------
Base (radix) of the machine (ibeta): 2
Number of base-ibeta digits in the mantissa (it): 24
Rounding mode (irnd): 5
Guard digits (ngrd): 0
Machine precision (machep): -23
Negative of machine precision (negep): -24
Number of bits in the exponent (iexp): 8
Minimum exponent (minexp): -126
Maximum exponent (maxexp): 128
Smallest positive floating-point number (xmin): 1.17549e-38
Largest positive floating-point number (xmax): 3.40282e+38
Machine precision (εM): 1.19209e-07
```

Machine Characteristics:

Base (radix) of the machine (ibeta): 2

Number of base-ibeta digits in the mantissa (it): 24

Rounding mode (irnd): 5

Guard digits (ngrd): 0

Machine precision (machep): -23

Negative of machine precision (negep): -24

Number of bits in the exponent (iexp): 8

Minimum exponent (minexp): -126

Maximum exponent (maxexp): 128

Smallest positive floating-point number (xmin): 1.17549e-38

Largest positive floating-point number (xmax): 3.40282e+38

Machine precision (εM): 1.19209e-07

The result that I get for single bit precision number:

Lower limit for single-bit precision number -1.4013e-45

Upper limit for single-bit precision number 1.70141e+38

Solution5:

rajeev_21111027_tut1.cpp ● test.cpp Release Notes: 1.81.0

Codes > rajeev_21111027_tut1.cpp > solution5()

```cpp
}
double my_cos(double x, long long n_max){
    double ans=0;
    long long fact_2i = 1;
    for (int i = 0; i <= n_max; i++)
    {
        double c = pow(-1, i%2);
        for (long long j = 0; j < 2*i; j++)
        {
            c*=x;
            c/=(j+1);
        }
        ans+=c;
        // cout<<c<<endl;
    }
    return ans;
}
void solution5(){
    double pi = 3.141592653, pi1 = 3.141592654;
    for (int i = 1; i < 10; i++)
    {
        cout<<setprecision(9);
        cout<<my_cos(-pi1+pi1*i/10, 100) << " " << cos(-pi1+pi*i/10) << endl;
        // break;
    }
}
int main()
{
    solution5();
    // solution2a();
    // solution2b();
```

For n_max = 100, the my_cos function was performing accurately up to 9 precision digits after decimal.

-0.951056516 -0.951056516
-0.809016995 -0.809016995
-0.587785253 -0.587785253
-0.309016995 -0.309016995
-2.05103454e-10 -7.0510347e-10
0.309016994 0.309016994
0.587785252 0.587785252
0.809016994 0.809016994
0.951056516 0.951056516

For n_max = 10000,

-0.951056516409238 -0.95105651644014
-0.809016994567838 -0.809016994685395
-0.587785252524778 -0.587785252767483
-0.309016994609025 -0.309016994989448
-2.05103453529017e-10 -7.05103469885517e-10
0.309016994218896 0.309016993648262
0.587785252192914 0.587785251626602
0.809016994326725 0.809016993856496
0.951056516282477 0.951056516004362

error for value theta = 3.141592654;

== -0.951056516409238(my_cos) -0.95105651644014(cos) = 3.0902×10^{-11};

Code link:
https://github.com/Rajeev-Gupta555/PHN-323/blob/master/rajeev_21111027_tut1.cpp

Machar header file link:

https://github.com/Rajeev-Gupta555/PHN-323/blob/master/machar.h