

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 = $1 + 0.f$;

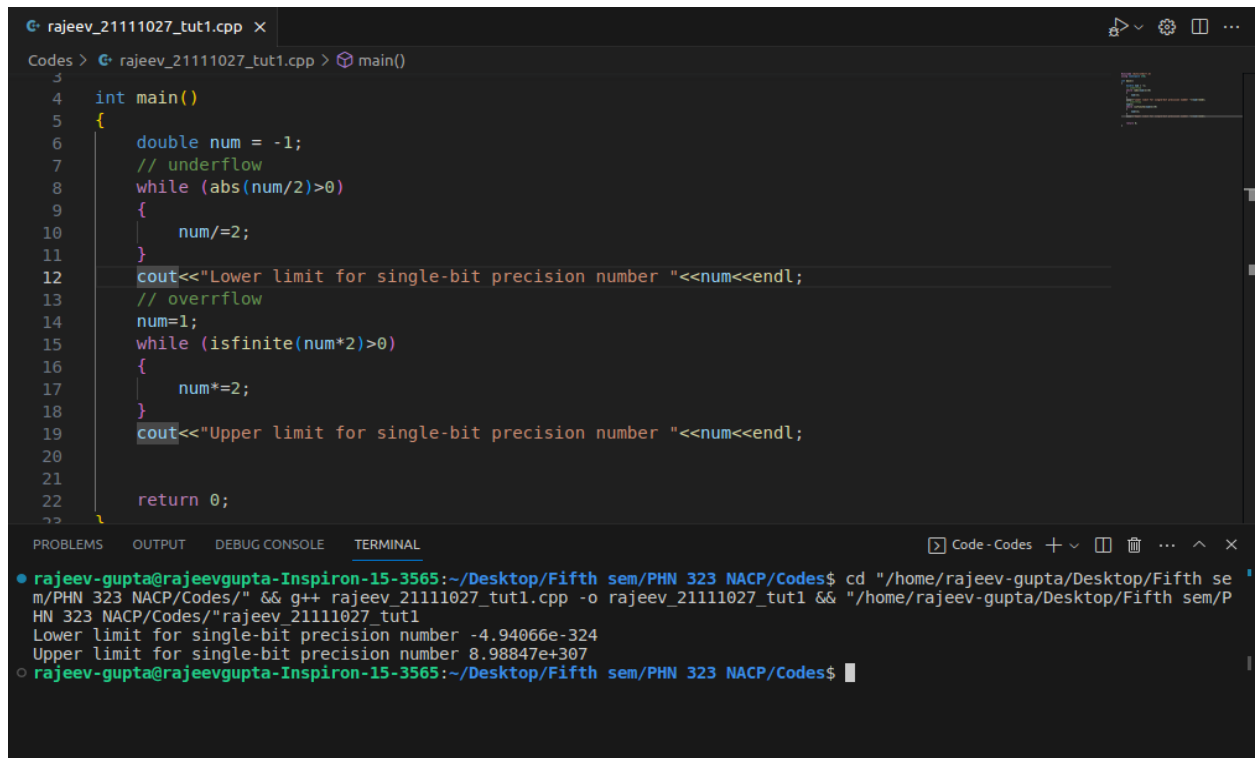
Mantissa = 1.625;

(d) Decimal value = $(-1)^{\text{sign}} * 2^{(\text{Exponent} - 127)} * 1.f$;

= $2^{-113} * 1.625$;

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

Solution2:



The screenshot shows a C++ IDE with a file named `rajeev_21111027_tut1.cpp`. The code defines a `main` function that calculates the lower and upper limits of single-bit precision numbers. The lower limit is found by repeatedly dividing `-1` by 2 until the absolute value is greater than 0. The upper limit is found by repeatedly multiplying 1 by 2 until the result is infinite. The program outputs the following results:

```
Lower limit for single-bit precision number -4.94066e-324
Upper limit for single-bit precision number 8.98847e+307
```

The terminal window shows the command used to compile and run the program:

```
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$
```

(a)

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

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

(b)

The image shows a C++ code editor with a file named `rajeev_21111027_tut1.cpp`. The code defines a `main()` function that calculates the lower and upper limits for single-bit precision numbers. The lower limit is found by repeatedly dividing `-1` by `2` until it underflows to zero. The upper limit is found by repeatedly multiplying `1` by `2` until it overflows to infinity. The results are printed using `cout`.

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main()
5 {
6     float num = -1;
7     // underflow
8     while (abs(num/2)>0)
9     {
10         num/=2;
11     }
12     cout<<"Lower limit for single-bit precision number "<<num<<endl;
13     // overflow
14     num=1;
15     while (isfinite(num*2)>0)
16     {
17         num*=2;
18     }
19     cout<<"Upper limit for single-bit precision number "<<num<<endl;
20 }
```

The terminal output shows the execution of the program:

```
rajeev-gupta@rajeevgupta-Inspiron-15-3565:~/Desktop/Fifth sem/PHN 323 NACP/Codes$ cd "/home/rajeev-gupta/Desktop/Fi
fth 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

The screenshot shows a C++ IDE with a file named `rajeev_21111027_tut1.cpp`. The code defines a `main()` function that calculates the upper and lower limits of an integer. The upper limit is found by doubling 1 until it overflows, and the lower limit is found by doubling -1 until it underflows. The output in the terminal shows the upper limit as 2147483647 and the lower limit as -2147483648.

```
Codes > rajeev_21111027_tut1.cpp > main()
34 // cout<<upper limit for double number <<endl;
35
36 int lower = -1;
37 int upper = 1;
38 // integers uses 2's complement, hence for a large values than the upperlimit of int
39 // the computer will consider it as a negative number.
40 while(upper>0){
41     // cout<<upper<<endl;
42     upper*=2;
43 }
44 cout<<"Upper limit for int number "<<upper-1<<endl;
45 // similiarly the negative value of int will become positive
46 // as the value is decreased further
47 while(1){
48     int check = lower*2;
49     if(check) lower=check;
50     else break;
51     // cout<<lower<<endl;
52 }
53 cout<<"Lower limit for int number "<<lower<<endl;
54
55 return 0;
56 }
```

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
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\$

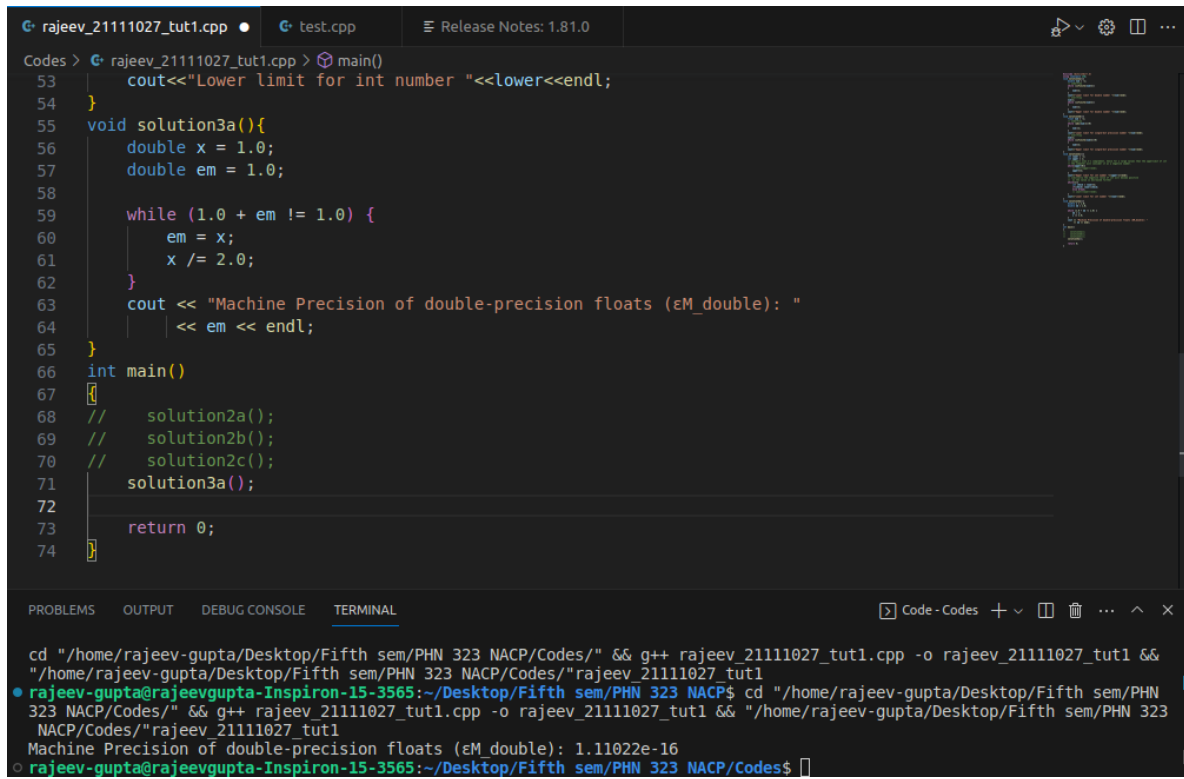
(c)

Upper limit for int number 2147483647

Lower limit for int number -2147483648

Solution 3:

(a)



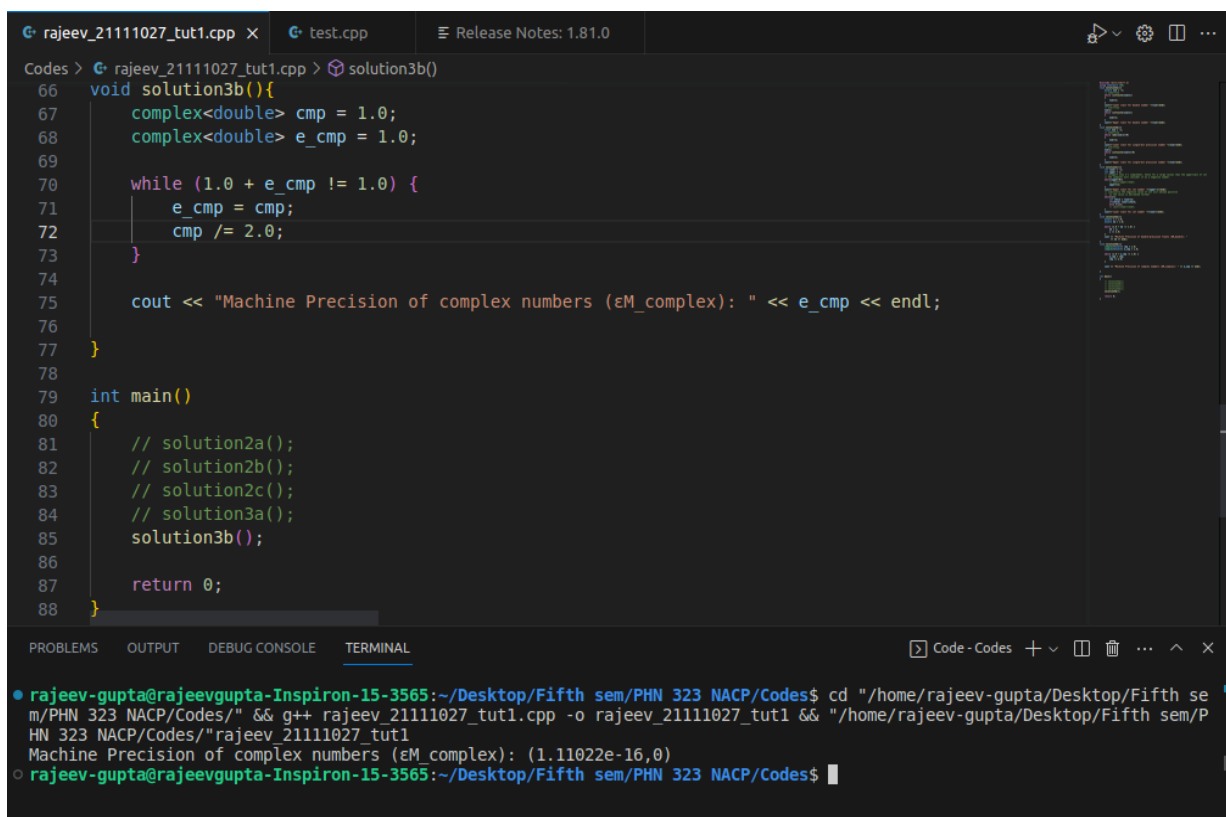
```
Codes > rajeev_21111027_tut1.cpp • test.cpp Release Notes: 1.81.0
53     cout<<"Lower limit for int number "<<lower<<endl;
54 }
55 void solution3a(){
56     double x = 1.0;
57     double em = 1.0;
58
59     while (1.0 + em != 1.0) {
60         em = x;
61         x /= 2.0;
62     }
63     cout << "Machine Precision of double-precision floats (εM_double): "
64         << em << endl;
65 }
66 int main()
67 {
68     // solution2a();
69     // solution2b();
70     // solution2c();
71     solution3a();
72
73     return 0;
74 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
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 (ϵ_{M_double}): 1.11022e-16

(b)



```
Codes > rajeev_21111027_tut1.cpp x test.cpp Release Notes: 1.81.0
66 void solution3b(){
67     complex<double> cmp = 1.0;
68     complex<double> e_cmp = 1.0;
69
70     while (1.0 + e_cmp != 1.0) {
71         e_cmp = cmp;
72         cmp /= 2.0;
73     }
74
75     cout << "Machine Precision of complex numbers (εM_complex): " << e_cmp << endl;
76
77 }
78
79 int main()
80 {
81     // solution2a();
82     // solution2b();
83     // solution2c();
84     // solution3a();
85     solution3b();
86
87     return 0;
88 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Code - Codes + -

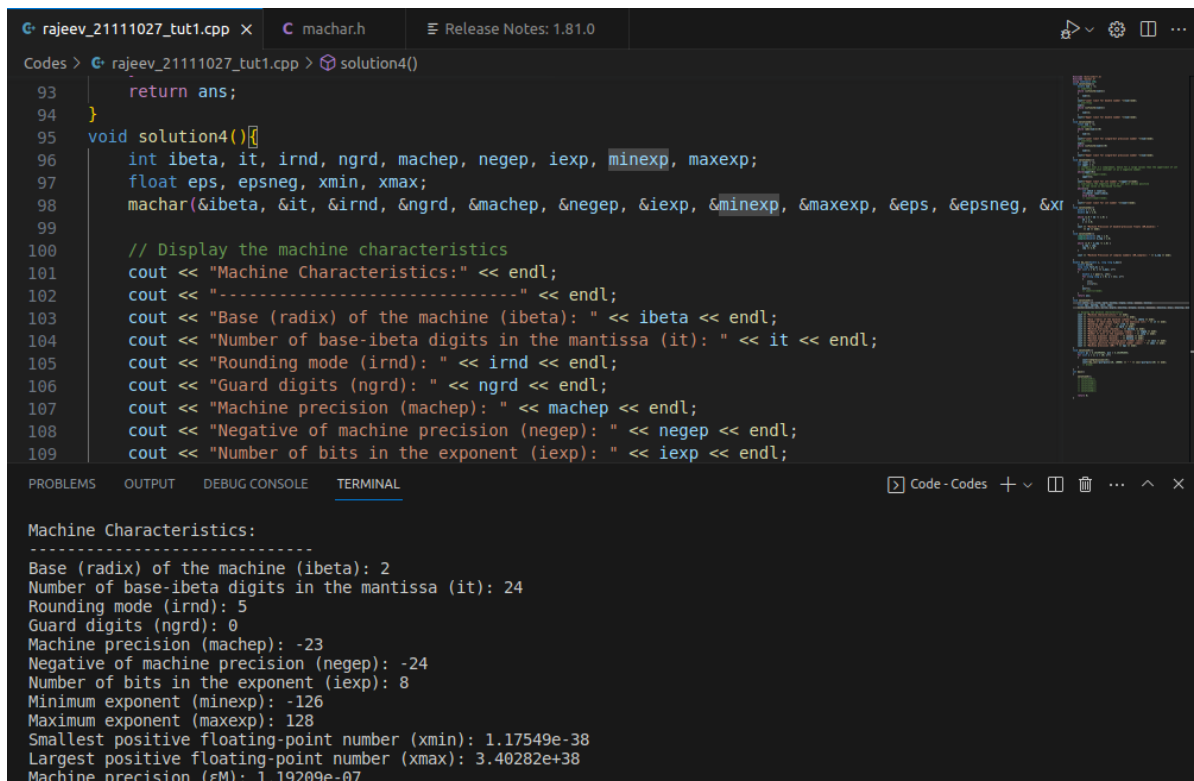
● rajeev-gupta@rajeevgupta-Inspiron-15-3565:~/Desktop/Fifth sem/PHN 323 NACP/Codes\$ cd "/home/rajeev-gupta/Desktop/Fifth sem/PHN 323 NACP/Codes/" && g++ rajeev_21111027_tut1.cpp -o rajeev_21111027_tut1 && "/home/rajeev-gupta/Desktop/Fifth sem/PHN 323 NACP/Codes/"rajeev_21111027_tut1

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:



```
Codes > rajeev_21111027_tut1.cpp > solution4()
93     return ans;
94 }
95 void solution4()
96 {
97     int ibeta, it, irnd, ngrd, machep, negep, iexp, minexp, maxexp;
98     float eps, epsneg, xmin, xmax;
99     machar(&ibeta, &it, &irnd, &ngrd, &machep, &negep, &iexp, &minexp, &maxexp, &eps, &epsneg, &xmin, &xmax);
100
101     // Display the machine characteristics
102     cout << "Machine Characteristics:" << endl;
103     cout << "-----" << endl;
104     cout << "Base (radix) of the machine (ibeta): " << ibeta << endl;
105     cout << "Number of base-ibeta digits in the mantissa (it): " << it << endl;
106     cout << "Rounding mode (irnd): " << irnd << endl;
107     cout << "Guard digits (ngrd): " << ngrd << endl;
108     cout << "Machine precision (machep): " << machep << endl;
109     cout << "Negative of machine precision (negep): " << negep << endl;
110     cout << "Number of bits in the exponent (iexp): " << iexp << endl;
111 }
```

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 > rajeiv_21111027_tut1.cpp > solution5()
77 }
78 double my_cos(double x, long long n_max){
79     double ans=0;
80     long long fact_2i = 1;
81     for (int i = 0; i <= n_max; i++)
82     {
83         double c = pow(-1, i%2);
84         for (long long j = 0; j < 2*i; j++)
85         {
86             c*=x;
87             c/=(j+1);
88         }
89         ans+=c;
90         // cout<<c<<endl;
91     }
92     return ans;
93 }
94 void solution5(){
95     double pi = 3.141592653, pil = 3.141592654;
96     for (int i = 1; i < 10; i++)
97     {
98         cout<<setprecision(9);
99         cout<<my_cos(-pil+pil*i/10, 100) << " " << cos(-pil+pi*i/10) << endl;
100         // break;
101     }
102 }
103 int main()
104 {
105     solution5();
106     // solution2a();
107     // 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>