Problem 1

Results Printed onto a table:

h	u	error
0.1	-0.49958347219741783	0.00041652780258211175
0.01	-0.4999958333473664	4.166652633530443e-6
0.001	-0.49999995832550326	4.167449668690537e-8
0.0001	-0.4999999969612645	3.0387354299499236e-9
1.0e-5	-0.50000115159321	1.1515932100691906e-6
1.0e-6	-0.499933427988708	6.657201129195434e-5
1.0e-7	-0.4996003610813205	0.0003996389186794458
1.0e-8	-1.1102230246251563	0.6102230246251563
1.0e-9	111.02230246251564	111.52230246251564
1.0e-10	0.0	0.49999999999999994
1.0e-11	0.0	0.49999999999999994
1.0e-12	0.0	0.49999999999999994
1.0e-13	1.1102230246251564e10	1.1102230246751564e10
1.0e-14	-1.1102230246251565e12	1.1102230246246565e12
1.0e-15	0.0	0.49999999999999994
1.0e-16	-1.1102230246251566e16	1.1102230246251566e16

Let me explain. Let the second order central difference be defined using the symbol $D^2_{\pm}[u]$, then the floating point error can be found by considering:

$$
\begin{aligned}
\mathrm{fl}(D^2_{\pm}[u]) &= \frac{u(x+h)(1+\epsilon_1) + 2u(x)(1+\epsilon_2) + (1+\epsilon_3)u(x-h)}{h} \\
&= D^2_{\pm}[u] + \frac{u(x+h)\epsilon_1 + 2u(x)\epsilon_2 + \epsilon_3 u(x-h)}{h} \\
&\approx \frac{3\epsilon u(x)}{h} \quad \text{When: } h >> 0
\end{aligned}
\tag{1.1}
$$

The truncation error is $\frac{h^2 u^{(4)}(x)}{12}$, the minimal amount of error is achieved when the truncation error and the round off error are close to each other, hence we can equate and solve for h:

$$
\begin{aligned}
\frac{h^2}{12}u^{(4)}(x) &= \frac{3\epsilon u(x)}{h} \\
\frac{h^3 u^{(4)}(x)}{12} &= 3\epsilon u(x) \\
h &= \sqrt[3]{\frac{36\epsilon u(x)}{u^{(4)}(x)}}
\end{aligned}
\tag{1.2}
$$

Letting $u(x)$ be $\sin(x)$ and $x = \pi/6$, we have $h = \sqrt[3]{36\epsilon}$, the case of using FLoat64, the best h is around $2e-5$. As we can se from the table, after the 5th row, the error rocketed, because the round off error takes off, and the round off error is proportional to $1/h$.

Problem 2

The results printed out to table is:

Schemes	Computed Results	Errors
φ_1 with h = 0.2	-0.49833555396895646	0.0016644460310435427
φ_1 with h = 0.1	-0.49958347219741783	0.00041652780258216726
φ_1 with h = 0.05	-0.4998958420134868	0.00010415798651319808
φ_2 with h = 0.2	-0.4999994449402383	5.55059761708776e-7
φ_2 with h = 0.1	-0.4999999652855098	3.471449022685036e-8
φ_3 with h = 0.2	-0.4999999999751922	2.480782246294666e-11

I derived the formula together with the order or accuracy by hands, which should come in later pages.

Code for Problem 2 and Problem 1

```
# Problem 1 and Problem 2 for the HW.

# To run the script, you need to add these packages using pkg to your Julia.
using Plots, Latexify, Logging, DataFrames


"""
    Second order second differential for fxn at x with h.
"""
function FiniteDiff2nd2ndAt(fxn, h, x)
    return (fxn(x + h) + fxn(x - h) - 2fxn(x))/h^2
end



"""
    one layer of richardson extrapolations
"""
function Richardson(fxn, h, x, i=1)
    if i == 1
        return FiniteDiff2nd2ndAt(fxn, h, x)
    end
    if i == 2
        return (4*Richardson(fxn, h/2, x, 1) - Richardson(fxn, h, x, 1))/3
    end
    if i == 3
        u = Dict()
        for h_ = [h, h/2, h/4, 0, -h/4, -h/2, -h]
            u[x + h_] = fxn(x + h_)
        end
        phi1(h) = h^(-2)*(u[x + h] - 2u[x] + u[x - h])
        phi2(h) = (4phi1(h/2) - phi1(h))/3
        phi3(h) = (16phi2(h/2) - phi2(h))/15
        return phi3(h)
    end
    error("not yet implemented")
end


function Problem1()
    x = pi/6
    Errors = Vector();
    Computed = Vector();
    hs = 10.0 .^ collect(-1:-1:-16)
    for h in hs
        FiniteDiff = FiniteDiff2nd2ndAt(sin, h, x)
        push!(Computed, FiniteDiff)
        push!(Errors, abs(-sin(x) - FiniteDiff))
    end
    @info "The error for problem 1 is: "
    fig = plot(log10.(hs), log10.(Errors))
    savefig(fig, "problem1_log_log.png")
```

```
    @info "The table is: "
    df = DataFrame(h=hs, u=Computed, error=Errors)
    latex = latexify(df, env=:mdtable)
    display(latex)
    latex = latexify(df, env=:tabular, latex=false)
    @info "The table in LaTeX is: "
    println(latex)
end


function Problem2()
    Results = Vector()
    push!(Results, "Computed Results")
    Schemes = Vector()
    push!(Schemes, "Schemes")
    Errors = Vector()
    push!(Errors, "Errors")
    h = 0.2
    push!(Results, Richardson(sin, h, x))
    push!(Schemes, "\$\\varphi_1\$ with h = $h")
    push!(Results, Richardson(sin, h/2, x))
    push!(Schemes, "\$\\varphi_1\$ with h = $(h/2)")
    push!(Results, Richardson(sin, h/4, x))
    push!(Schemes, "\$\\varphi_1\$ with h = $(h/4)")
    push!(Results, Richardson(sin, h, x, 2))
    push!(Schemes, "\$\\varphi_2\$ with h = $(h)")
    push!(Results, Richardson(sin, h/2, x, 2))
    push!(Schemes, "\$\\varphi_2\$ with h = $(h/2)")
    push!(Results, Richardson(sin, h, x, 3))
    push!(Schemes, "\$\\varphi_3\$ with h = $(h)")
    append!(Errors, abs.(Results[2:end] .+ .5))
    TableToPrint = hcat(Schemes, Results, Errors)
    @info "The table for problem 2 is: "
    latexify(TableToPrint, env=:tabular, latex=false) |> print

end

Problem1()
Problem2()
```

It's on github if you want to view the source code directly just email me and I will add you to the git repo.