EX 1

// Create two error variables, one called ErrInvalidValue and the other

// called ErrAmountTooLarge. Provide the static message for each variable.

// Then write a function called checkAmount that accepts a float64 type value

// and returns an error value. Check the value for zero and if it is, return

// the ErrInvalidValue. Check the value for greater than $1,000 and if it is,

// return the ErrAmountTooLarge. Write a main function to call the checkAmount

// function and check the return error value. Display a proper message to the screen.

package main

import (

"errors"

"fmt"

)

var (

// ErrInvalidValue indicates the value is invalid.

ErrInvalidValue = errors.New("Invalid Value")

// ErrAmountTooLarge indicates the value beyond the upper bound.

ErrAmountTooLarge = errors.New("Amount To Large")

)

func main() {

// Call the function and get the error.

if err := checkAmount(0); err != nil {

switch err {

// Check if the error is an ErrInvalidValue.

case ErrInvalidValue:

fmt.Println("Value provided is not valid.")

return

// Check if the error is an ErrAmountTooLarge.

case ErrAmountTooLarge:

fmt.Println("Value provided is too large.")

return

// Handle the default error.

default:

fmt.Println(err)

return

}

}

// Display everything is good.

fmt.Println("Everything checks out.")

}

// checkAmount validates the value passed in.

func checkAmount(f float64) error {

switch {

// Is the parameter equal to zero.

case f == 0:

return ErrInvalidValue

// Is the parameter greater than 1000.

case f > 1000:

return ErrAmountTooLarge

}

return nil

}

**EX2**

// Create a custom error type called appError that contains three fields, err error,

// message string and code int. Implement the error interface providing your own message

// using these three fields. Implement a second method named temporary that returns false

// when the value of the code field is 9. Write a function called checkFlag that accepts

// a bool value. If the value is false, return a pointer of your custom error type

// initialized as you like. If the value is true, return a default error. Write a main

// function to call the checkFlag function and check the error using the temporary

// interface.

package main

import (

"errors"

"fmt"

)

// appError is a custom error type for the program.

type appError struct {

err error

message string

code int

}

// Error implements the error interface for appError.

func (a \*appError) Error() string {

return fmt.Sprintf("App Error[%s] Message[%s] Code[%d]", a.err, a.message, a.code)

}

// Temporary implements behavior about the error.

func (a \*appError) Temporary() bool {

return (a.code != 9)

}

// temporary is used to test the error we receive.

type temporary interface {

Temporary() bool

}

func main() {

if err := checkFlag(false); err != nil {

switch e := err.(type) {

case temporary:

fmt.Println(err)

if !e.Temporary() {

fmt.Println("Critical Error!")

}

default:

fmt.Println(err)

}

}

}

// checkFlag returns one of two errors based on the value of the parameter.

func checkFlag(t bool) error {

// If the parameter is false return an appError.

if !t {

return &appError{errors.New("Flag False"), "The Flag was false", 9}

}

// Return a default error.

return errors.New("Flag True")

}