Control   Cont		Haskell Platform + Semigro	uns + Semiaro	inoids + Fither	Most Frequen	fly used types (	tynoclassos						
March   10			SemiGroup	Monoids	Functor	Alt	Plus	Apply	Applicative		Bind	Monad	
The content of the	"Base"												
Section   Sect													
Bay   Property Color	Bool	All D.MONOID											
Corp. Sections 1. 1		Any D.MONOID											
Manual Processor   Manual Proc		ByteString D.BS,D.BS.L											
The projects review Mail 2 Nat 2 Nat 2 Nat 2 Nat 3 Nat													
No. 2	Num	Product a D.MONOID		Num a									
Mark		Min a D.SEMI	Ord a										
Designation	Ord	May a D.SEMI	Ordo										
Company   Comp													
Compared		[a] D.LIST											
Seg_ ***	0	D.LIST				х	х				х	х	Х
Segretary   Part		Seg a D.SEQ	×	Y	Х			X	X	*			
Monthing a first		Seq D.SEQ	^	_ ^	х	x	x	x	x	x	x	x	х
May be a provided at the control of the control o			х										
Manuface   Service   Control   Con				DList a		*		×		×	X		×
March Representation		Set a D.SET	Ord a	Ord a									
Map   Ame	Int												
Marga   1.000				а									
Manufact		Map k a D.MAP	Ord k	Ord k		Ordic	Ordik	Ordik			Ordik		
Matthigs   1		IntMap a D.IMAP	x	x	X	Olu K	Olu K	Olu K			Olu K		
Monte   March   Marc	Int	IntMap D.IMAP			х	х	х	х			х		
Major   Profession   Professi													
Mappe   Mapp													
Maybe   Maybe   Semigroup						y Y	Y	×		×			Y
Moyebo   Options a ****   Semigroup   Se	Maybe	Maybe a D.M	Semigroup a	Monoid a	^								
	Maybe	Option D.SEMI	Com!	Com:	х	х	х	х	х	х	х	х	х
List a professor   X		First a D.M	Semigroup a										
Ether a	Maybe	Last a D.M											
Either a ****		First a D.SEMI											
Either a   Section   Sec		Either a D.EITH	^		Х	х		х	X		х	х	
		Either a b D.EITH	х										
Mayee						Alt m	Plus m			Alternative m			MonadPlus m
	Mayhe						Bind m, Monad		Functor m,	Functor m,	Bind m, Monad		
District   Control   Con	-	-											
	0						Applicative m				m		
State   m   Semigroup   Monad m   Semigroup   Monad m   Semigroup   Monad m   Monad													
Either   Error r					Functor m	Alt m	Plus m		Applicative m				
Ether		StateT s m C.M.T.S			Functor m	Alt m	Plus m	Bind m			Bind m	Monad m	MonadPlus m
RWST r w sm = 64.596	Either e	ErrorT e m <sup>C.M.T.ERR</sup>			Functor m					Monad m,			
Either   E		RWST r w s m <sup>C.M.T.RWS</sup>			Functor m	Alt m		Bind m,	Monoid w,	Monoid w,	Bind m,	Monoid w,	Monoid w,
Selling	Fither					Monad m,			Monad m,	MonadPlus m Monad m,	-		Monad m,
Parsec's u m		EitherT e m a C.M.T.E	Semigroup m			Semigroup e							
WrappedApplicative 1972   Monoid m   Monoi													
WrappedArrow a b C-APP   Arrow a Monad m   M		WrappedMonoid m D.SEMI	Monoid m	Monoid m									
Margoward of Supplementaries   Arrow a   Arr					Manad m				Manad m	ManadDlua m	Monad m		
Arrow/Ronad a CARR											Monad m		
					Allow a	AllowPlus a	AllowPlus a	Allow a	Allow a	ArrowPlus a			Arrow Apply o
ST s C M					Arrow a				Arrow a	ArrowPlus a		ArrowApply a	ArrowPlus a
STM						X	Х	X			X		
ReadPrec		STM			х				х			х	
Tuples  (a, b) Semigroup a, Monoid a, Semigroup b, Monoid b, b, Mo													
Tuples  (a, b, c) Semigroup b, Monoid b, Monoid b, Semigroup b, Monoid b, Mo													
Semigroup b   Semigroup b   Semigroup b   Monoid b   Semigroup a   Monoid a   Semigroup m   Semigr	Tueler		Semigroup a,	Monoid a,									
(-) a Semigroup b Monoid b Semigroup a Semigr	Tuples	(a, b, c)	Semigroup b,	Monoid b,									
Semigroup b   Monoid b					Х			Semigroup a	Monoid a		Semigroup m		
Const a b CAPP Compose f g Das Semigroup a X Semigroup m Monoid m Static f a Das Semigroup m Monoid m Applicative f Applicative f Applicative f Applicative f Applicative g Data May potential part granulation and gr		a → b Endo a D.MONOID											
Semigroup m   Monoid m   Static a Dass   Functor f   Alt f   Plus f   Apply f   Applicative		(→) a						х	х		х	х	
Static f a poss   Functor f   Alt f   Plus f   Apply f   Applicative f   Applicative f   Applicative f   Applicative f   Applicative g   App			Semigroup a					Semigroup m	Monoid m				
Functor g   Functor g   Applicative g   Appl		Static f a D.SS			Functor f	Alt f	Plus f		Applicative f				
Note 1: Typeclasses in Haskell imply laws, not (necessarily) semantics   Note 2: While a uniform semantic / behaviour would be nice to have, most 'class laws' were found insufficient to provide this, and yet no additional laws were specified (hysterical raisins)   C.APP		Compose f g D.SS											<u> </u>
Note 1: Typeclasses in Haskell imply laws, not (necessarily) semantics  Note 2: While a uniform semantic / behaviour would be nice to have, most 'class laws' were found insufficient to provide this, and yet no additional laws were specified (hysterical raisins)  C.APP Control.Applicative base D.BS Data.ByteString bytestring D.M Data.Maybe, Prelude base C.M.ST Control.Monad.STT base D.D.LIST Data.Dilst dilist D.MONOID Data.Monoid, Prelude base C.M.T.E Control.Monad.Trans.Either transformers D.EITH Data.Either, Prelude base D.O.R.D Data.Dilst Data.Semigroupoids D.SEMI Data.Semigroup semigroupoids D.SEMI Data.Semigroupoids D.SEMI Data.Semigroup semigroupoids D.SEMI Data.Semigroupoids D.SEMI D		Product f a D.F.C			Functor f,			Apply f. Annly a	Applicative f,	Applicative f,	Bind f. Bind a		
Note 2: While a uniform semantic / behaviour would be nice to have, most 'class laws' were found insufficient to provide this, and yet no additional laws were specified (hysterical raisins)  C.APP Control.Applicative base D.BS Data.ByteString bytestring D.M Data.Maybe, Prelude base D.BS. Control.Arrow base D.BLS. Data.ByteString.Lazy bytestring D.MAP Data.Map containers D.MAP Data.Dilist dilist D.MONOID Data.Monoid, Prelude base D.MSC Data.Delist Data.Delist D.MAP Data.Delist D.MAP Data.Map containers D.MAP Data.Delist D.MAP Data.Delist D.MAP Data.Map containers D.MAP Data.Delist D.MAP Data.Deli	Note 1		oly laws not (no	ecessarily) com				FF 7 9	Applicative g	Applicative g	g		
C.ARP Control.Applicative base D.BS Data.ByteString bytestring D.M Data.Maybe, Prelude base C.ARR Control.Arrow base D.BS.L Data.ByteString.Lazy bytestring D.MAP Data.Map containers dlist D.MONOID Data.Monoid, Prelude base D.D.LIST Data.Dilst dlist D.MONOID Data.Monoid, Prelude base D.MONOID Data.Monoid, Prelude base D.M.T.C Ontrol.Monad.Trans.Error transformers D.F.A Data.Functor.Apply semigroupoids D.SEMI Data.Semigroup semigroups D.S.EMI Data.Semigroup S.D.EMI Data.Del		,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,		• •						ditional laws were specified (hysterical raising			ns)
C.M.ST Control.Monad.ST base C.M.T.E Control.Monad.Trans.Either transformers C.M.T.ER Control.Monad.Trans.Either transformers D.F.H Data.Functor.Apply semigroupoids semigroupoids D.SEM Data.Semigroup containers C.M.T.I Control.Monad.Trans.List transformers D.F.C Data.Functor.Compose transformers C.M.T.I Control.Monad.Trans.List transformers D.F.C Data.Functor.Compose transformers D.F.C Data.Functor.Compose transformers D.F.C Data.Functor.Compose transformers D.F.C Data.Functor.Compose transformers D.F.C Data.Functor.Individual Control.Monad.Trans.Maybe transformers D.F.I Data.Functor.Individual Control.Monad.Trans.Reader transformers D.F.I Data.Functor.Individual Control.Monad.Trans.Reader transformers D.F.I Data.HashMap unordered-containers D.F.I Data.Text text C.M.T.R Control.Monad.Trans.State transformers D.H.SET Data.HashSet unordered-containers D.T.L Data.Text.Lazy text C.M.T.R Control.Monad.Trans.State transformers D.H.SET Data.Individual Control.Monad.Trans.Control.Monad.Trans.State Transformers D.H.SET Data.Individual Control.Monad.Trans.Control.Monad.Trans.State Transformers		Control.Applicative		base				bytestring		D.M Data.Maybe,			base
C.M.T.ER Control.Monad.Trans.Either transformers C.M.T.ERR Control.Monad.Trans.Either transformers transformers D.F.A Data.Functor.Apply semigroupoids D.SEMI Data.Semigroup semigroupoids D.SEMI Data.Semigroup Semigroupoids D.SEMI Data.Semigroup Semigroupoids D.SEMI Data.Semigroup Semigroupoids D.SEMI Data.Semigroupoids D.SEMI Da					D.DLIST Data.Dlist D.EITH Data.Either, Prelude D.F.A Data.Functor.Apply D.F.B Data.Functor.Bind D.F.C Data.Functor.Comp		Prelude Apply	dlist e base semigroupoids semigroupoids transformers		D.MONOID D.ORD D.SEMI		Prelude	
C.M.T.I Control.Monad.Trans.Identity transformers C.M.T.L Control.Monad.Trans.List transformers C.M.T.M Control.Monad.Trans.Maybe transformers D.F.C Data.Functor.Compose transformers D.S.E Data.Set containers D.S.E Data.Set Co	C.M.T.E	Control.Monad.Trans.Either		transformers							Data.Ord, Prel	ude	base
C.M.T.L Control.Monad.Trans.List transformers C.M.T.M Control.Monad.Trans.List transformers C.M.T.M Control.Monad.Trans.Maybe transformers D.F.L Data.Functor.Identity transformers D.SS Data.Semigroupoids semigroupoids C.M.T.RWS Control.Monad.Trans.Reader transformers D.HMAP unordered-containers D.T.L Data.Text text C.M.T.RWS Control.Monad.Trans.State transformers D.HMSE Data.HashMap unordered-containers D.T.L Data.Text.Lazy text C.M.T.RWS Control.Monad.Trans.State transformers D.I.SET Data.InSet containers D.T.E Data.Tree containers D.ATTO Data.Attoparsec attoparsec D.I.SET Data.List, Prelude base T.PARSEC Text.Parsec.Prim parsec			ı										
C.M.T.R Control.Monad.Trans.Reader transformers C.M.T.RWS Control.Monad.Trans.RWST transformers D.HMAP Data.HashMap unordered-containers D.T. Data.Text text text C.M.T.RWS Control.Monad.Trans.RWST transformers D.HMSET Data.HashSet unordered-containers D.T.L Data.Text.Lazy text containers C.M.T.W Control.Monad.Trans.Writer transformers D.HMAP Data.IntMap containers D.T.REE Data.Tree containers D.HMAP Data.List, Prelude base T.PARSEC Text.Parsec.Prim parsec	C.M.T.L	Control.Monad.Trans.List		transformers			Compose			D.SET Data.Set			containers
C.M.T.RWS Control.Monad.Trans.RWST transformers C.M.T.S Control.Monad.Trans.State transformers D.HSET Data.HashSet unordered-containers D.T.L Data.Text.Lazy text containers D.HSET Data.Instance Containers D.T.C Data.Tree containers D.T.C Data.Tree Control.Monad.Trans.Writer transformers D.J.SET Data.InsSet containers D.HSET Data.InsSet Control.Monad.Trans.Writer D.J.SET Data.InsSet Containers D.HSET Data.Tree Container								transformers				upoid.*	
C.M.T.S Control.Monad.Trans.State transformers D.IMAP Data.IntMap containers D.TREE Data.Tree containers C.M.T.W Control.Monad.Trans.Writer transformers D.ISET Data.InsSet containers D.ISET Data.InsSet containers S.IO System.IO, Prelude base D.ISET Data.List, Prelude base T.PARSEC Text.Parsec.Prim parsec	C.M.T.RWS	Control.Monad.Trans.RWST transformers		D.HSET	Data.HashSet		unordered-containers		D.T.L	T.L Data.Text.Lazy		text	
D.ATTO Data.Attoparsec attoparsec D.LIST Data.List, Prelude base T.PARSEC Text.Parsec.Prim parsec								containers			E Data.Tree		
D.L.NE Data.List.NonEmpty semigroups					D.LIST	LIST Data.List, Prelude base							
					D.L.NE	Data.List.NonE	Empty	semigroups					

	Rinary operation semanti	c / Mempty 'meaning' / Additional Laws										
	Class:	SemiGroup	Monoids		Alt Data.Functor.Alt	Plus	Apply Data.Functor.Apply	Applicative	Alternative	Bind	Monad	Monad Plus
"Base"	Type	Data.Semigroup	Data.f	mempty	Data.Functor.Alt	Data.Functor.Plus Zero	Data.Functor.Apply	Control Applicative	Control.Applicative	Data.Functor.Bind >>-,join	>>=, return	mplus,mzero
Dase	Ordering	Choice	Choice	EQ	\.'\	2610	·	, puie	\ r,empty	,joiii	, return	IIIpius,IIIzero
	0	None	None	()								
Bool Bool	All	Combine Combine	Combine Combine	True False								
Word8	ByteString	Combine	Combine	Empty								
Char	Text	Combine	Combine	Empty								
Num	Sum a	Combine	Combine	0								
	Product a Min a	Combine Choice	Combine Choice	1 maxBound								
Ord	Max a	Choice	Choice	minBound								
	Dual a	~ a	~ a	~ a								
0	[a]	Both	Both	Empty	Both	Empty	<*>	Both	х	X	х	Left Dist.
Ш	ZipList				Botti	Linpty	<*>	Both	*	_ ^	^	Leit Dist.
	Seq a	Both	Both	Empty								
	Seq NonEmpty a	Both			Both	Empty	ар	X	х	x	х	X
	NonEmpty	DOUT			Both		ар	x		x	x	
	DList a		Both	Empty			- 1	Both	Both		х	х
	Set a	Both	Both	Empty								
Int	IntSet HashSet a	Both Both	Both Both	Empty Empty								
	Map k a	Both	Both	Empty								
	Map k	<b>.</b>	5."		Both	Empty	()			Ord k		
Int	IntMap a IntMap	Both	Both	Empty	Both	Empty	()			x		
	HashMap k a	Both	Both	Empty	DOUT	Linpty	()					
	HashMap k			1.5								
	Tree Maybe				Choice	Empty	apDefault	X	Choice	X X	X X	Left Catch
Maybe	Maybe a	Combine	Combine	Empty	CHOICE	Lilipty	appelault	^	CHOICE	_ ^	^	Len Galon
Maybe	Option				Choice	Empty	<*>	х	Choice	х	х	х
iviaybe	Option a	Combine	Combine	Empty								
Maybe	First a Last a		Choice Choice	Empty Empty								
	First a	Choice	Onoice	2ptg								
	Last a	Choice										
	Either a Either a b	Choice		~ Empty	Choice		()	x		×	Х	
	Identity	Choice		Linpty			<*>	x		x	x	
	IdentityT m				Combine	~ m	<.>	Applicative m	Alternative m	Bind m	Monad m	MonadPlus m
Maybe	MaybeT m				Choice	Empty	apDefault	Functor m, Monad m	Functor m, Monad m	Bind m, Monad m	Monad m	Monad m
п	LintT m				Combine	- Franti	( )			Bind m, Monad	Monad m	Monad m
0	ListT m				Combine	Empty	()	Applicative m	Applicative m	m		
	ReaderT e m				Combine	~ m	()	Applicative m Monoid w,	Alternative m Monoid w,	Bind m Bind m,	Monad m Monoid w,	MonadPlus m Monoid w,
	WriterT w m				Combine	~m	()	Applicative m	Alternative m	Semigroup w	Monad m	MonadPlus m
	StateT s m				Combine	~ m	apDefault	Functor m,	Functor m,	Bind m	Monad m	MonadPlus m
								Monad m	MonadPlus m Functor m,			
Either e	ErrorT e m				Choice	~ Empty	apDefault	Functor m, Monad m	Monad m,	Bind m, Monad m	Monad m, Error e	Monad m, Error e
		-						Monoid w,	Error e Monoid w,			
	RWST r w s m				Combine	~ m	apDefault	Functor m,	Functor m,	Bind m, Semigroup w	Monoid w, Monad m	Monoid w, MonadPlus m
								Monad m	MonadPlus m	Serrigioup w	Worlau III	
Either	EitherT e m				Both?		()	Monad m, Monoid e	Both	Monad m	Monad m	Monad m, Monoid e
Either	EitherT e m a	Choice										,
	Parser DATTO			X				X	X		X	X
	ParsecT s u m TPARSEC WrappedMonoid m	~ m	~ m	x ~ m				X	Х		Х	Х
	WrappedApplicative f	<u> </u>			~ f	~ f	~ f					
	WrappedMonad m				~ f	~ m	~ m	~ m	~ m	~ m		
	WrappedArrow a b				~ a	~ a	~ a	~ a	ArrowZero a, ArrowPlus a			
	ArrowMonad a							~ a	~ a		ArrowApply a	ArrowApply a,
	IO				Chair-	000-	<*>		ď			ArrowPlus a
	ST s				Choice	error	\">	X		X	X X	
	STM							х	Х		х	Left Catch
	ReadP ReadPrec	-						X	X		X	X
	(a, b)	~a, ~b	~a, ~b	~a, ~b				Х	Х		Х	X
Tuples	(a, b, c)	~a, ~b, ~c	~ a, ~ b, ~c	~ a, ~ b, ~c								
	(,) a		L	L			()	Monoid a		Semigroup m		
	a → b Endo a	~ b Neither	~ b Neither	~ b Neither								
	(→) a	14010101	14010101	14010101			<*>	х		х	Х	
	Const a b	~ a						NA				
	Const m Static f a				~ f	Plus f	<> ()	Monoid m Applicative f				
					- 1	1 105 1	()	Aplicative f,	Alternative f,			
	Compose f g							Applicative g	Alternative g			
	Product f g (D.F.P)						()	Applicative f, Applicative g	Applicative f, Applicative g	Bind f, Bind g		
	Cokleisli w a						()	Х	y		Х	

Choice
Combine
Binary operation chooses one of the values
Both [Lists/Seq/Nempty]
Both [Sets/Maps]
Binary operation combines both values
Binary operation chooses all possible outcomes, thus combining them
Binary operation combines both values, choosing from left when conflicts arise
None of the above concepts makes sense in this context
as a
Empty
mempty/zero value reflects empty container