ha ter 1

In r du i n

1.1 Th L int rfac

We have implemented an ML interface to Xlib, the industry standard C interface for X at the lowest level, and which is widely used as the basis for many toolkits. We provide all the major function groups, so that this interface can be used to implement fully functional complex applications. We also provide a of geometric functions for handling points and
and a of functions for pe6onsminglogical operations on plane masks and pixel values.

Xlib is now widely documented, with many goo and programming manuals available.
We provide our version of the Xlib manual with ML signatures and types, and a more
functional style to the programming interface.

We provide ML example programs to show the functionality of the ML interface to Xlib. These
examples range from simple linewing applications through to colour examples and a mini
text editor showing how to with The full s.3214 0 Td (an7.7474 0 Td (of)Tj 11.9785 0 Td (the)Tj 17.805 0 Td (struc4 0 T

Datatypes are used where possible so that arguments are strongly typed and pattern matching may be used for returned values, this is especially useful for pattern matching the di erent sub-types of events. Abstypes are only used for the X resource types which have no meaningful textual representation.

The functions have been made more functional. Where an Xlib function modi ed its arguments, this has been changed so that the function returns new, modi ed copies of the arguments. Where values were passed in partially lled-in structures with OR-ed bit masks, now the programmer uses constructors to make the list of v

ha ter 2

Fun i n R f r n

.1 olours, Pix ls and R B valu s

2.1.1 And, Or, Xor, Not, >>, <<

Types:

```
infix And Or Xor >> <<

val Not: int -> int
val And: int * int -> int
val Or:  int * int -> int
val Xor: int * int -> int
val >> : int * int -> int
val << : int * int -> int
```

Description:

These functions provide useful arithmetic operations on ints representing pixel values and plane masks, which, in X, are unsigned 32-bit quantities. The least signi cant bits of these quantities are on the right, an

And, Or and Xor perform b

Not performs bitwise negatio

a >> b returns a shifted b bi shifted b bits to the left, whe

If negative values, or values greate exception Range is raised.

2.1.2 BlackPixel, WhitePixel

Types:

```
val BlackPixel: unit -> int
val WhitePixel: unit -> int
```

#RRGGBB	(8 bits each)
#RRRGGGBBB	(12 bits each)
#RRRRGGGGBBBB	(16 bits each)

The R, G, and B represent single hexadecimal digits (both uppercase

Syntax:

```
        val depth = DefaultDepth() ;
```

2.2.5 XCreateColormap, XCopyColormapAndFree, XFreeColormap,

Syntax:

```
XInstallColormap cmap ;
XUninstallColormap cmap ;
val cmaps = XListInstalledColormaps w ;
```

Arguments:

cmap	Speci es the colormap.
w	Speci es the window that determines the screen.

Description:

The **XInstallColormap** function installs the speci ed colormap for its associated screen. All windows associated with this colormap immediately display with true colours. You associated the windows with this colormap when you created them by calling **XCreateWindow**, **XCreateSimpleWindow**, **XChangeWindowAttributes**, or **XSetWindowColormap**.

If the speci ed colormap is not already an installed colormap, the X server generates a **ColormapNotify** event

property	Speci es the property atom.
stdmaps	Speci es the XStandardColormaps to be used
maps	Returns the XStandardColormap

Argument Type:

```
datatype XStandardColormap = XStandardColormap of { colormap:  Colormap,
                                                    redMax:    int,
                                                    redMult:   int,
                                                    greenMax:  int,
                                                    greenMult: int,
                                                    blueMax:   int,
                                                    blueMult:  int,
                                                    basePixel: int,
                                                    visual:    Visual }
```

Argument Description:

The colormap member is the colormap created by the X CreateColormap function. The redMax, greenMax, and blueMax members give the maximum red, green, and blue values, respectively. Each colour coe cient ranges from zero to its max, inclusive. For example, a common colormap allocation is 3/3/2 (3 planes for red, 3 planes for green, and 2 planes for blue). This colormap would have redMax = 7, greenMax = 7, and blueMax = 3. An alternate allocation thately

The **XGetRGBColormaps** function returns the RGB colormap de nitions stored in the speci ed property on the named window. If the property exists, is of type **RGB_COLOR_MAP**, is of format 32, and is long enough to contain a colormap de nition (if the visual is not present, **XGetRGBColormaps** assumes the default visual for the screen on which the window is located), **XGetRGBColormaps** returns the ic 0 s

hotspot Speci es the x and y coordinates, which

.4 isplay Sp cifications

2.4.1 AllPlanes

Types:

```
val AllPlanes: int
```

Syntax:

```
val planeMask = AllPlanes ;
```

Description:

AllPlanes is a value with all bits set to 1 and is suitable for use in a plane mask argument to a function.

2.4.2 BitmapBitOrder

Types:

```
val BitmapBitOrder: unit -> ImageOrder
```

Argument Type:

```
datatype ImageOrder = LSBFirst | MSBFirst
```

Syntax:

```
val order = BitmapBitOrder() ;
```

Description:

The BitmapBitOrder function returns LSBFirst or MSBFirst to indicate whether the leftmost bit in the bitmap as displayed on the screen is the least or most signi cant bit in the bytes combitmaptdata.

2.4.3 BitmapPthead

ypes:

```
val BitmapPad: unit -> int
```

ntax:

```
val pad = BitmapPad() ;
```

escription:

The BitfunapPtheadumber of bits that each scanline mustbe padded.

2.4.4 BitmapUnit

Types:

```
val BitmapUnit: unit -> int
```

Syntax:

```
val scanline = BitmapUnit() ;
```

Description:

The BitmapUnit function returns the size of a bitmap's scanline unit in bits.

2.4.5 ByteOrder

Types:

```
val ByteOrder: unit -> ImageOrder
```

Argument Type:

```
datatype ImageOrder = LSBFirst | MSBFirst
```

Syntax:

```
val order = ByteOrder ;
```

Description:

The ByteOrder function speci es the val6otl6edbyte order for images for each scanline unit in XY

2.4.7 ColormapExists,

2.4.9 DefaultVisual

T

2.4.12 DisplayPlanes

Types:

```
val DisplayPlanes: unit -> int
```

Syntax:

```
val planes = DisplayPlanes() ;
```

Description:

The DisplayPlanes function returns the depth of the root window of the screen.

2.4.13 DisplayString

Types:

```
val DisplayString: unit -> string
```

Syntax:

```
val s = DisplayString() ;
```

Description:

The DisplayString function returns the string that was passed to XOpenDisplay when the current display was

2.4.15 DoesSaveUnders

Types:

```
val DoesSaveUnders: unit -> bool
```

Syntax:

```
val su = DoesSaveUnders() ;
```

2.4.18 MaxCmapsOfScreen

Types:

```
val MaxCmapsOfScreen:  unit -> int
```

Syntax:

```
val n = MaxCmapsOfScreen() ;
```

Description:

The **MaxCmapsOfScreen** function returns the maximum number of installed colormaps supported by the screen.9552 Tf -405.142 -29.1992 Td -36.9638 Td (2.4.19)Tj 47.752 0 Td (NoColormap,)Tj 82.

2.4.24 VendorRelease

Types:

```
val VendorRelease: unit -> int
```

Syntax:

```
val n = VendorRelease() ;
```

Description:

The VendorRelease function returns a numb
server.

This function uses these GC components: function, plane-mask, subwindow-mode, graphics-exposures, clip-origin, and clip-mask.

The XCopyPlane function uses a single bit plane of the speci ed souece rectangle combined with the speci ed GC to modify the speci ed rectangle of dest. The drawables must have the

Description:

XDrawArc draws a single circular or elliptical arc, and **XDrawArcs** draws multiple circular or elliptical arcs. Each arc is speci ed by

endpoin

2.5.5 XDrawLine, XDrawLines, XDrawSegments

Types:

```
val XDrawLine:     Drawable -> GC -> XPoint -> XPoint -> unit
val XDrawLines:    Drawable -> GC -> XPoint list -> CoordMode -> unit
val XDrawSegments: Drawable -> GC -> (XPoint * XPoint) list -> unit
```

Syntax:

```
XDrawLine d gc point1 point2 ;
XDrawLines d gc points mode ;
XDrawSegments d gc segments ;
```

Arguments:

d	**Speci es the drawable.**
gc	**Speci es the GC.**
mode	**Speci es the coordinate mode. You can pass CoordModeOrigin or CoordModePrevious.**
points	**Speci es a list of points.**
segments	

2.5.6 XDrawPoint, XDrawPoints

Types:

```
val XDrawPoint:  Drawable -> GC -> XPoint -> unit
val XDrawPoints: Drawable -> GC -> XPoint list -> CoordMode -> unit
```

Syntax:

```
XDrawPoint d gc point ;
XDrawPoints d gc points mode ;
```

Arguments:

d **Speci es the dra**

.n

For fonts de ned with 16-bit linear indexing and used with **XDrawString**, each 8-bit c

area	Speci es the bounding rectangle of the area. The x and y coordinates, which are relative to the origin of the drawable, specify the upper-left corner of the bounding rectangle. The width and height are the major and minor axes of the arc.
angle1	Speci es the start of the arc relative to the three-o'cd4he

The ll-rule of the GC controls the lling behavior of self-intersecting polygons.

This function uses these GC components: foreground, background, tile, stipple, tile-stipple-origin, function, plane-mask, ll-style, ll-rule, subwindow-mode, clip-origin, and clip-mask.

For each arc, XFillArc or XFillArcs lls the region closed by the in nitely thin path described by the speci ed arc and, depending on the arc-mode speci ed in the GC, one or two line segments. For ArcChord, the single line segment joining the endpoints of the arc is used. For ArcPieSlice, the two line segments joining the endpoints of the arc with the center point are used. XFillArcs lls the arcs in the same order as the list. For any given arc, XFillArc and XFillArcs do not draw a pixel more than once. If regions intersect, the intersecting pixels are drawn multiple times.

Both functions use these
t components: foreground, background, tile, stipple, tile-stipple-origin, function, plane-mask, ll-style, arc-mode, subwindow-mode, clip-origin, and clip-mask.

"Not a window"	**Attempt to use a pixmap Drawable as a window use a window Drawable**
"Not a pixmap"	**Attempt to use**

```
val ShiftDown:   Modifier list -> bool
val ControlDown: Modifier list -> bool
```

Syntax:

```
ShiftDown   modifiers
ControlDown modifiers
```

Arguments:

modi ers	**Speci es the modi ers from a key event**

Description:

The ShiftDown convenience function returns true if ShiftMask is in the modi ers list, and false otherwise. This indicates if the Shift key was pressed when the key event was generated.

The ControlDown convenience function returns true if ControlMask is in the modi ers list, and false otherwise. This indicates if the Control key was pressed when the key event was generated.

2.7.3 XLookupString, NoSymbol

Types:

```
val XLookupString: int -> Modifier list -> (string * int)
val NoSymbol:      int
```

Syntax:

```
val (string,keysym) = XLookupString keycode modifiers ;
```

Arguments:

keycode	**Speci es the keycode from a key**

Description:

modi ers **Speci es the modi ers from a key ev**

Descriptithe string for that combination the keysym for that combination

DescriXLookupString translates a key ev

Description:8.2408 0 Td (to)Tj 13.1654 0 Td (a)Tj 9.31829 0 Td (KeySym)Tj 40.837 0 Tc

ard interpretation

than the current X server time. Otherwise, the last-focus-change time is set to the speci ed time (**CurrentTime** is replaced by the current X server time). **XSetInputFocus** causes

error handling

Description:

The XTranslateCoordinates function takes the srcPoint coordinates relative to the source window's origin and returns these coordinates to dstPoint relative to the destination window's origin. If XTranslateCoordinates returns zero, srcWindow and dest-

. Fonts

Types:

```
val CharLBearing:   XCharStruct         XCharStruct XCharStruct XCharStruct
```

Window are on di erent scree
a mapped child cf destWindo
value NoDrawable.

```
val FSAllCharsExist: XFontStruct -> bool
val FSDefaultChar:   XFontStruct -> int
val FSMinBounds:     XFontStruct -> XCharStruct
val FSMaxBounds:     XFontStruct -> XCharStruct
val PSPerChar:       XFontStruct -> XCharStruct list
val FSAscent:        XFontStruct -> int
val FSDescent:       XFontStruct -> int

val FSMinWidth:  XFontStruct -> int
val FSMaxWidth:  XFontStruct -> int
val FSMinHeight: XFontStruct -> int
val FSMaxHeight: XFontStruct -> int
```

Argument Typ

If the perChar list is empty, all glyphs between the rst and last character indexes inclusive have the same information, as given by both minBounds and maxBounds.tan-336.09.349 -13.9465 Td

irecCharse31Tf693623 0 rd (irighr) Tj43433623 0andther

Arguments:

fs	Speci es the XFontStruct to use.
string	Speci es the character string.
bigChars	Speci es the character string as a list [illegible]

Description:

```
val MakeRect:  XPoint -> XPoint -> XRectangle
val SplitRect: XRectangle  -> (XPoint * XPoint)
```

Syntax:

```
val (topLeft,bottomRight) = SplitRect r ;
val r = MakeRect corner1 corner2 ;
```

Description:

MakeRect constructs an XRectangle givemo points correspondaing pair of opposite corners of the rectangle. This is useful when the order of the t o points is not known, for example when dragging a rubber-banded box on the screen.

SplitRect returns the pair of points correspondingthe top-left and bottom-right corners of the XRectangle. It will always be the case that `left <= right` **and** `top <= bottom`**.**

2.9.6 NegativePoint

Types:

```
val NegativePoint: XPoint -> XPoint
```

Description:

NegativePoint negates both the x and y coordinates of the point. This is equivalent to re ecting about the x axis and the y axis.

2.9.7 OutsetRect, O setRect, IncludePoint

Types:

```
val OutsetRect:   int -> XRectangle -> XRectangle
val OffsetRect:   XRectangle -> XPoint -> XRectangle
val IncludePoint: XPoint -> XRectangle -> XRectangle
```

Description:

`OutsetRect n R` **takes rectangle R and expands its area by n units i**
Typically n is positive and this function is used to expand areas to in
the same width all around. With a negativ

©Abstr

manufacturers' line drawing hardware, which may run many times

gc Speci es the GC

2.10.18 XSetState

Types:

```
val XSetState: GC -> int -> int -> GCFunction -> int -> unit
```

Syntax:

```
XSetState gc foreground background function planeMask ;
```

Arguments:

background	**Speci es the background pixel.**
foreground	**Speci es the foreground**

Sp

2.10.22 XSetTSOrigin

T

2.11.2 VisualRedMask, VisualGreenMask, VisualBlueMask

Types:

```
val VisualRedMask:   Visual -> int
val VisualGreenMask: Visual -> int
val VisualBlueMask:  Visual -> int
```

Syntax:

```
val redMask   = VisualRedMask   visual ;
```

bytesPerLine **Sp**

The

destArea **Speci es coordinates relative to the origin of the dra**

are returned for regions of the window that are obscured b

Description:

The **XInternAtom** function returns the atom identi er associated with the speci ed name string. If onlyIfExists

```
                                | PropertyVisual      of Visual list
                                | PropertyWindow      of Drawable list
                                | PropertyWMHints     of XWMHint list
                                | PropertyWMSizeHints of XWMSizeHint list
                                | PropertyWMIconSizes of (XRectangle *
                                                          XRectangle *
                                                          XRectangle) list
```

Properties:

WM_CLIENT_MACHINE	**The string name of the machine on which the client application is running.**
WM_COMMAND	**The command and arguments, separated by ASCII n**

perform the conversion then you call **XSendSelectionNotify** with property set to zero to indicate that the conversion failed. If the conversion was successful then the requestor will read the value from the property on the window, and will delete the property to indicate that the transfer has been completed.

.13 Scr n Sav r

2.13.1

XSetScreenSaver enables the screen saver. An interval of 0 disables the random-pattern motion. If no input from devices (keyboard, mouse, and so on) is generated for the speci ed number of timeout seconds once the screen saver is enabled, the screen saver is activated.

For each screen, if blanking is preferred and the hardware supports video blanking, the screen simply goes blank. Otherwise, if either exposures are allowed

Syntax:

```
val      default         =   XGetDefault           program        option         ;
```

Arguments:

option	Speci es the option name.
program	Speci es the program name.

Description:

XGetDefault returns the value for the program and option entry in the user's defaults database. If **XGetDefault** fails then exception **XWindows** is raised with "XGetDefault failed" .

.16 indows

2.16.1 XCreateWindow, XCreateSimpleWindow

Types:

```
val      XCreateWindow:                  Drawable          ->     XPoint          ->     XRectangle          ->
                                         int      ->     int      ->     WindowClass             ->     Visual          ->
                                         XSetWindowAttributes                        list      ->     Drawable

val      XCreateSimpleWindow:                      Drawable          ->     XPoint          ->     XRectangle          ->
                                                   int      ->     int      ->     int      ->     Drawable
```

Syntax:

```
val      window         =   XCreateWindow              parent        point        area
                            borderWidth              depth        class        visual          attributes            ;

   val      window         =   XCreateSimpleWindow                   parent        point        area
                               borderWidth              borderPixel              backgroundPixel                ;
```

Arguments:

attributes	Speci es the initial values for the window's attributes.
backgroundPixel	Speci es the bac 5 0 ordWidth(X2)1393182dc7(-2(1)59)(nj)tbickgro)u)9)6)2Ry2556264 Tf dil09

c

```
val XGetGeometry: Drawable -> (Drawable * XPoint * XRectangle * int * int)

val XGetWindowAttributes: Drawable -> XWindowAttributes
```

Syntax:

```
val (root,position,size,borderWidth,depth) = XGetGeometry w ;
val attributes = XGetWindowAttributes w ;
```

Arguments:

d **Speci es the drawable, which can be a windo**

ta9:Ddr6.7327 0 Td (ot)Tj /R92 9.96264 Tf 80.3194 0 Td (Returns)Tj 37.8172 0 Td (the)Tj 17.1247 0

changes	**Speci es a list of XWindowChanges.**
w	**Speci es the wo w to be recon gured.**
borderWidth	**Speci es the wodth of the wo w border.**
area	**Speci es the interior dimensions of the wo w.**
origin	**Speci es the x and y coordinates, which de ne the new location of the top-left pixel of the wo w's b**

Syntax:

```
XRaiseWindow w ;
XLowerWindow w ;
XCirculateSubwindows w direction ;
XCirculateSubwindowsDown w ;
XCirculateSubwindowsUp w ;
XRestackWindows windows ;
```

Arguments:

direction	Speci es the direction (up or down) that you want to circulate the window. You can pass **RaiseLowest** or **LowerHighest**.
w	Speci es the window.
windows	Speci es the list of windows to be restacked.

Argument Type:

```
datatype CirculateDirection = RaiseLowest | LowerHighest
```

Description:

The **XRaiseWindow** function raises the speci ed window to the top of the stack so that no sibling window obscures it. If the windows are regarded children are not a ected. This is a convenience function equivalent to **XCirculateSubwindows** with **RaiseLowest** speci ed.

The **XCirculateSubwindowsDown** function lowers the highest mapped child of the speci ed window that partially or completely occludes another child. Completely

2.16.12 XUnmapWindow, XUnmapSubwindows

Types:

```
val XUnmapWindow:     Drawable -> unit
val XUnmapSubwindows: Drawable -> unit
```

Syntax:

```
XUnmapWindow w ;
XUnmapSubwindows w ;
```

Arguments:

w	**Speci es the window.**

Description:

The XUnmapWindow function unmaps the speci ed window and causes the X server to generate an UnmapNotify event. If the speci ed window is already

Syntax:

```
    XSetWMClass w class ;
    val class = XGetWMClass w ;
```

Arguments:

class	**Speci es the class names for the window**
w	**Speci es the window**

Properties:

WM_CLASS	**Set by application programs to allow window and session managers to obtain the application's resources from the resource database.**

_CLASS property on the speci ed window. XGetWM-Class returns the WM_CLASS property on the speci ed window.

2.17.4 XSetWMClientMachine, XGetWMClientMachine

Types:

```
    val XSetWMClientMachine: Drawable
```

CLIENT

2.17.5 XSetWMColormapWindows, XGetWMColormapWindows

Types:

```
val XSetWMColormapWindows: Drawable
```

w	Speci es the window.
commands	Speci es the list of strings.

Description:

The **XSetWMCommand** function sets the **WM_COMMAND** property on the speci-
ed window. T

Description:

The **XSetWMIconName** convenience function performs a **XSetProperty** on the **WM_ICON_NAME** property. The **XSetWMIconName** function sets the name to be displayed in a window's icon.

The **XGetWMIconName** convenience function performs an **XGetTextProperty** on the **WM_ICON_NAME** property. The **XGetWMIconName** function returns the name be displayed in the speci ed window's icon. If it succeeds, it returns the name, otherwise, if no icon name has been set for the window, it raises exception **XWindows** with "Xb

The **XGetWMSizeHints** function returns the

ha ter 5

E

```
ColormapNotify of { sendEvent: bool,
                    window:    Drawable,
                    colormap:  Colormap,
                    new:       bool,
                    installed: bool }
```

Description:

The window member is set to the window whose associated colormap is changed, installed, or

3.7 onfi ur R qu st

Types:

```
datatype StackMode = Above | Below | TopIf | BottomIf | Opposite ;

ConfigureRequest of { sendEvent:   bool,
                      parent:      Drawable,
                      window:      Drawable,
                      position:    XPoint,
                      size:        XRectangle,
                      borderWidth: int,
                      above:       Drawable,
                      detail:      StackMode }
```

memb isset the 26 bor (d 2123
•ec54 -2d6 7345773 T set (the) Tj 16.5578 0 T
order •ec54 -2d6 7346.5 T9d (The) Tj 19.8754 0 T
w•ec54 -2d6 7345.5 T0 Td (set) (b)1 0 Td (w) Tj 9.9426 0 Td

borej b7.64729 0 Td (bsrecon reRed.j /A 1 Tf 0.1 0 0 -0.1 90.
order wi to

Description:

The parent member is set to the parent window. The window member is set to the window whose size, position, border width, and/or 7Tj 7j 2ar

3.17 MapNotify

Types:

```
MapNotify of { sendEvent:        bool,
               event:            Drawable,
               window:           Drawable,
               overrideRedirect: bool }
```

Description:

The event member is set either to the window that was mapped or to its parent, depending on whether **StructureNotifyMask** or **SubstructureNotifyMask** was selected. The window member is set to the window that was mapped. The overrideRedirect member is set to the override-redirect attribute of the window. Window manager clients normally should ignore this window if the override-redirect attribute is true, because these events usually are generated from pop-ups, which override structure cj 84.614.6056 0 Td (tol.5)Tj /A 1 Tf 0.1 0 0 -0.1

```
ReparentNotify of { sendEvent:      bool,
                    event:          Drawable,
                    windowparent:   Drawable, windowoverrideRedirect: XPoint},
```

Description:

The event member is set either to the reparented window or to the old or the new parent, depending on whether **StructureNotifyMask** or **SubstructureNotifyMask** was selected. The window member is set to the window that was reparented. The parent member is set to the new parent window. The position member is set to the reparented window's coordinates relative to the new parent window's origin and de nes the upper-left outer corner of the reparented window. The overrideRedirect member is set to the override-redirect attribute of the window sdeci ed by the window member. Window manager clients normally should ignore this window if the overrideRedirect member is true.

3. 1 R siz R qu st

Types:

```
ResizeRequest of { sendEvent: bool,
                   window:    Drawable,
                   size:      XRectangle }
```

Description:

The window member is set to the window whose size another client attempted to change. The size member is set to the inside size of the window, excluding the border.

3. S l ction l ar

Types:

```
SelectionClear of { sendEvent: bool, window:    Drawable,
                                     selection: timi}ection:
```

Description:

The window member is set to the window losing ownership of the selection. The selection member is set to the selection atom. The timi member is set to the last change timi recorded for the selection. The owner member is the window that was sdeci ed by the current owner in its **XSetSelectionOwner** call.

3. 3 S l ctionNotify

Types:

```
SelectionNotify of { sendEvene: bool,
                     requestor: Drawable,
                     selection: int,
                     targee:    int,
                     property:  int,
                     time:      int }
```

Description:

The requestor member il set to the window associated with the

Description:

The event member is set either to the unmapped window or to its parent, depending on whether **StructureNotifyMask** or **SubstructureNotifyMask** was selected. This is the window used by the X server to report the event. Th ha ter 4

r l Err r

4.1 BadPixmap

Description:

A value for a Pixmap does not name a de ned Pixmap. ML type-checking should avoid this error.

4.13 BadR qu st

Description:

This should never occur in Xlib since only standard requests are made.

4.14 BadValu

Description:

Some numeric value falls outside the range of values accepted by the request.

XAllocColorCells	**Number of colours must be positive and planes must be non-negative.**
XAllocColorPlanes	**Number of colours must be positive, and reds, green and blues must be non-negative**
XFreeColors	**Speci ed pixel is not a valid index into the colormap.**
XBell	**Percent must be ~100 to 100.**
XResizeWindow	**Window width must be non-zero.**
XCopyPlane	**Plane must have one bit set to 1, and specify an existing plane.**
XCreateGlyphCursor	**Source char and mask char must exist in the font.**
XSetDashes	**Dash elements must be positive and less than 256.**
XCreatePixmap	**Speci ed width must be non-zero, and depth must be supported.**
XSetScreenSav	**areaues**/V8Td (XSetDashes)Tjtor/R92 9.9

Ind x