

X Window System Protocol
MIT X Consortium Standard
X Version 11, Release 5

Robert W. Scheifler
Massachusetts Institute of Technology
Laboratory for Computer Science

X Window System is a trademark of M.I.T. **X Window System Protocol**

Copyright © 1986, 1987, 1988 Massachusetts Institute of Technology

MIT X Consortium Standard
X Version 11, Release 5
Permission to use, copy, modify, and distribute this document for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies. All rights reserved. No part of this document may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or by any information storage and retrieval system, without the prior written permission of MIT. The copyright notice and this permission notice are retained, and that the name of M.I.T. not be used in advertising or publicity pertaining to this document without specific, written prior permission. M.I.T. makes no representations about the suitability of this document or the protocol defined in this document for any purpose. It is provided "as is" without express or implied warranty.

Robert W. Scheifler
Massachusetts Institute of Technology
Laboratory for Computer Science

Acknowledgments

The primary contributors to the X11 protocol are:

Dave Carver (Digital HPW)
Branko Gerovac (Digital HPW)
Jim Gettys (MIT/Project Athena, Digital)
Phil Karlton (Digital WSL)
Scott McGregor (Digital)
Dan Rason (Digital UEC)
David Rosenthal (Sun)
Dave Whitham (Digital UEC)

X Window System Protocol
X Consortium Standard
X Version 11, Release 5

Permission to copy, modify, and distribute this document for any purpose and without fee is hereby granted, provided that the copyright notice and this permission notice are retained in all copies. This document may be used in advertising or publicity pertaining to this document without specific, written prior permission. M.I.T. makes no representations about the suitability of this document or the protocols it describes for any particular purpose. M.I.T. expressly disclaims any express or implied warranty.

Susan Angebrannt (Digital)
Raymond Drewry (Digital)
Todd Newman (Digital) Robert W. Scheifler

The invited reviewers who provided useful input are:
Massachusetts Institute of Technology
Laboratory for Computer Science

Andrew Cherenon (Berkeley)
Burns Fisher (Digital)
Dan Garfinkel (HP)
Leo Hourvitz (Next)
Brock Krizan (HP)
David Laidlaw (Stellar)
Dave Mellinger (Interleaf)
Ron Newman (MIT)
John Ousterhout (Berkeley)
Andrew Palay (ITC CMU)
Ralph Swick (MIT)
Craig Taylor (Sun)
Jeffery Vroom (Stellar)

Thanks go to Al Mento of Digital's UEG Documentation Group for formatting this document.

This document does not attempt to provide the rationale or pragmatics required to fully understand the protocol or to place it in perspective within a complete system.

The protocol contains many management mechanisms that are not intended for normal applications. Not all mechanisms are needed to build a particular user interface. It is important to keep in mind that the protocol is intended to provide mechanism, not policy.

Robert W. Scheifler
Massachusetts Institute of Technology
Laboratory for Computer Science

1. Protocol Formats

Request Format

Acknowledgments

Every request contains an 8-bit major opcode and a 16-bit length field expressed in units of four bytes. Every request consists of four bytes of a header (containing the major opcode, the length field, and a data byte) followed by zero or more additional bytes of data. The length field defines the total length of the request, including the header. The length field in a request must equal the minimum length required to contain the request. If the specified length is smaller or larger than the required length, an error is generated. Unused bytes in a request are not required to be zero.

Major opcodes 128 through 255 are reserved for extensions. Extensions are intended to contain multiple requests. Requests typically have an additional minor opcode encoded in the "spare" data byte in the request header. However, the placement and interpretation of this minor opcode and of all other fields in extension requests are not defined by the core protocol. Every request of a given connection is implicitly assigned a sequence number, starting with one, that is used in replies, errors, and events. (C)

The implementation of this protocol is provided by the X Window System. The copyright notice and permission to use this document for any purpose and without fee is hereby granted, provided that the copyright notice and this permission notice are retained and that the name of MIT is used in advertising or publicity pertaining to this document without specific, written prior permission. M.I.T. makes no representations about the suitability of this document or the protocol for use in any particular circumstances. M.I.T. disclaims any liability for any damage or loss resulting from the use of the information contained herein.

The implementation of this protocol is provided by the X Window System. The copyright notice and permission to use this document for any purpose and without fee is hereby granted, provided that the copyright notice and this permission notice are retained and that the name of MIT is used in advertising or publicity pertaining to this document without specific, written prior permission. M.I.T. makes no representations about the suitability of this document or the protocol for use in any particular circumstances. M.I.T. disclaims any liability for any damage or loss resulting from the use of the information contained herein.

The implementation of this protocol is provided by the X Window System. The copyright notice and permission to use this document for any purpose and without fee is hereby granted, provided that the copyright notice and this permission notice are retained and that the name of MIT is used in advertising or publicity pertaining to this document without specific, written prior permission. M.I.T. makes no representations about the suitability of this document or the protocol for use in any particular circumstances. M.I.T. disclaims any liability for any damage or loss resulting from the use of the information contained herein.

Reply Format

Every reply contains a 32-bit length field expressed in units of four bytes. Every reply consists of 32 bytes followed by zero or more additional bytes of data, as specified in the length field.

Unused bytes are not guaranteed to be zero. Every reply also contains the least-significant 16 bits of the sequence number of the corresponding request. The invited reviewers who provided useful input are:

Andrew Cherenon (Berkeley)
Burns Fisher (Digital)
Dan Garhinkel (HP)
Leo Hourvitz (Next)
Brock Khzan (HP)
David Ladlaw (Stellar)
Dave Mellinger (Interleaf)
Ron Newman (MIT)
John Ousterhout (Berkeley)
Andrew Palay (ITC CMU)
Ralph Swick (MIT)
Craig T. Sobot (Typecraft)
Jeffrey Senders (Sun)

Events are 32 bytes long. Every event contains an 8-bit error code. The most-significant bit in this code is set if the event was generated from a SendEvent request. Event codes 64 through 127 are reserved for extensions, although the core protocol does not define a mechanism for selecting interest in such events. Thanks go to all members of the Digital Group for formatting this document.

Every request of a given connection is implicitly assigned a sequence number of the last request issued by the client that was (or is currently being) processed by the server. This document does not attempt to provide the rationale or pragmatics required to fully understand the protocol or to place it in perspective within a complete system.

The protocol contains many management mechanisms that are not intended for normal applications. Not all mechanisms are needed to build a particular user interface. It is important to keep in mind that the protocol is intended to provide a mechanism, not policy.

The syntax {...} encloses a set of alternatives. The set of structure components.

In general, TYPES are in uppercase and Alternative Values are capitalized.

Requests in section 9 are described in the following format:

RequestName
arg1: type1
...
argN: typeN
=>

2. Syntactic Conventions

The syntax {...} encloses a set of alternatives. The set of structure components.

In general, TYPES are in uppercase and Alternative Values are capitalized.

Requests in section 9 are described in the following format:

RequestName
arg1: type1
...
argN: typeN
=>

1. Protocol Formats: typeL

Request Format resultM: typeM Acknowledgments

Every request contains an 8-bit major opcode and a 16-bit length field expressed in units of four bytes. Every request consists of four bytes of a header (containing the major opcode, the length field, and a data byte) followed by zero or more additional bytes of data. The length field defines the total length of the request, including the header. The length field in a request must equal the minimum of the request to contain the request. If the specified length is smaller or larger than the requested length, an error is generated. Unused bytes in a request are not required to be zero.

Although a request may still be a Window System Protocol, multiple requests may be sent to a single server. Requests typically have an additional minor opcode encoded in the data byte of the request header. However, the placement and interpretation of this minor opcode varies between fields in extension requests are not defined by the core protocol. Every request of a given connection may implicitly assigned a sequence number, starting with one, that is used in replies, errors, and events. (EO)

The implementation or user may provide useful information to express or implied warranty. The implementation or user may provide useful information to express or implied warranty.

Reply Format

Every reply consists of a 32-bit length field expressed in units of four bytes. Every reply consists of 32 bytes followed by zero or more additional bytes of data, as specified in the length field. Unused bytes in a reply are not guaranteed to be zero. Every reply also contains the least-significant 16 bits of the sequence number of the corresponding request.

3. Common Types

The invited reviewers who provided comments of Technology

Name Format Value Laboratory for Computer Science

Andrew Cherenon (Berkeley)

Error reports are 32 bytes long. Every error includes an 8-bit error code. Error codes 128 through 255 are reserved for extensions. Every error also includes the major and minor opcodes of the failed request and the least-significant 16 bits of the sequence number of the request. For the following errors (see section 4), the failing resource ID is also returned: Colormap, Cursor, Drawable, Font, GContext, IDChoice, Pixmap, and Window. For Atom errors, the failing atom is returned. For Value errors, the failing value is returned. Other core errors return no additional data. Unused bytes within an error are not guaranteed to be zero.

BITMAP, BITMASK, and LISTOFVALUE are somewhat special. Variants of these requests contain arguments of the form: value-mask: BITMASK

Event Format

Events are 32 bytes long. Unused bytes within an event are not guaranteed to be zero. Every event contains an 8-bit event code. The most-significant bit in this code is set if the event was generated from a SendEvent request. Event codes 64 through 127 are reserved for extensions. Arguments are to be provided; each such argument is assigned a unique bit position. The representation of the BITMASK will typically contain more bits than there are defined arguments. The unused bits in the BITMASK must be zero (or the server generates a Value error). The value list contains one value for each bit set to 1 in the mask, from least-significant to most-significant bit in the mask. Each value is represented in four bytes to build a 32-bit value, or up to only the most significant byte if the procedure is indicated in the mechanism specified by the name of the protocol.

The protocol contains many mechanisms that maintain the mask. Each value is represented in four bytes to build a 32-bit value, or up to only the most significant byte if the procedure is indicated in the mechanism specified by the name of the protocol.

The rest of this procedure is indicated in the mechanism specified by the name of the protocol. The name of the protocol is indicated in the mechanism specified by the name of the protocol.

The name of the protocol is indicated in the mechanism specified by the name of the protocol.

The name of the protocol is indicated in the mechanism specified by the name of the protocol.

The name of the protocol is indicated in the mechanism specified by the name of the protocol.

The name of the protocol is indicated in the mechanism specified by the name of the protocol.

The name of the protocol is indicated in the mechanism specified by the name of the protocol.

The name of the protocol is indicated in the mechanism specified by the name of the protocol.

The name of the protocol is indicated in the mechanism specified by the name of the protocol.

The name of the protocol is indicated in the mechanism specified by the name of the protocol.

The name of the protocol is indicated in the mechanism specified by the name of the protocol.

ATPMtocol Formats: 32-bit value (top three bits guaranteed to be zero)

VISUALID ... 32-bit value (top three bits guaranteed to be zero)

Request Format result 32-bit value (top three bits guaranteed to be zero)

BYTE Every request contains an 8-bit major opcode and a 16-bit length field expressed in units of four bytes. Every request consists of four bytes of a header (containing the major opcode, the length field, and a data byte) followed by zero or more additional bytes of data. The length field defines the total length of the request, including the header. The length field in a request must equal the minimum number of bytes present in the request.

INT16 16-bit signed integer. If the specified length is smaller or larger than the minimum number of bytes present in the request, the request has no reply (it is asynchronous).

CARD16 16-bit unsigned integer. If the specified length is smaller or larger than the minimum number of bytes present in the request, the request has no reply (it is asynchronous).

TIME 32-bit unsigned integer. If the specified length is smaller or larger than the minimum number of bytes present in the request, the request has no reply (it is asynchronous).

DOUBLE 64-bit floating point value. If the specified length is smaller or larger than the minimum number of bytes present in the request, the request has no reply (it is asynchronous).

POINT 16-bit signed integer. If the specified length is smaller or larger than the minimum number of bytes present in the request, the request has no reply (it is asynchronous).

RECTANGLE 32-bit unsigned integer. If the specified length is smaller or larger than the minimum number of bytes present in the request, the request has no reply (it is asynchronous).

STRING 32-bit unsigned integer. If the specified length is smaller or larger than the minimum number of bytes present in the request, the request has no reply (it is asynchronous).

KEYMAP 32-bit unsigned integer. If the specified length is smaller or larger than the minimum number of bytes present in the request, the request has no reply (it is asynchronous).

NAME 32-bit unsigned integer. If the specified length is smaller or larger than the minimum number of bytes present in the request, the request has no reply (it is asynchronous).

POINT 16-bit signed integer. If the specified length is smaller or larger than the minimum number of bytes present in the request, the request has no reply (it is asynchronous).

POINT 16-bit signed integer. If the specified length is smaller or larger than the minimum number of bytes present in the request, the request has no reply (it is asynchronous).

POINT 16-bit signed integer. If the specified length is smaller or larger than the minimum number of bytes present in the request, the request has no reply (it is asynchronous).

POINT 16-bit signed integer. If the specified length is smaller or larger than the minimum number of bytes present in the request, the request has no reply (it is asynchronous).

POINT 16-bit signed integer. If the specified length is smaller or larger than the minimum number of bytes present in the request, the request has no reply (it is asynchronous).

POINT 16-bit signed integer. If the specified length is smaller or larger than the minimum number of bytes present in the request, the request has no reply (it is asynchronous).

POINT 16-bit signed integer. If the specified length is smaller or larger than the minimum number of bytes present in the request, the request has no reply (it is asynchronous).

POINT 16-bit signed integer. If the specified length is smaller or larger than the minimum number of bytes present in the request, the request has no reply (it is asynchronous).

POINT 16-bit signed integer. If the specified length is smaller or larger than the minimum number of bytes present in the request, the request has no reply (it is asynchronous).

POINT 16-bit signed integer. If the specified length is smaller or larger than the minimum number of bytes present in the request, the request has no reply (it is asynchronous).

POINT 16-bit signed integer. If the specified length is smaller or larger than the minimum number of bytes present in the request, the request has no reply (it is asynchronous).

POINT 16-bit signed integer. If the specified length is smaller or larger than the minimum number of bytes present in the request, the request has no reply (it is asynchronous).

POINT 16-bit signed integer. If the specified length is smaller or larger than the minimum number of bytes present in the request, the request has no reply (it is asynchronous).

POINT 16-bit signed integer. If the specified length is smaller or larger than the minimum number of bytes present in the request, the request has no reply (it is asynchronous).

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

...near-linear increasing rumps in each primary.

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

...near-linear increasing rumps in each primary.

[REDACTED]

[REDACTED]

...near-linear increasing rumps in each primary.

[REDACTED]

...near-linear increasing rumps in each primary.

[REDACTED]

...near-linear increasing rumps in each primary.

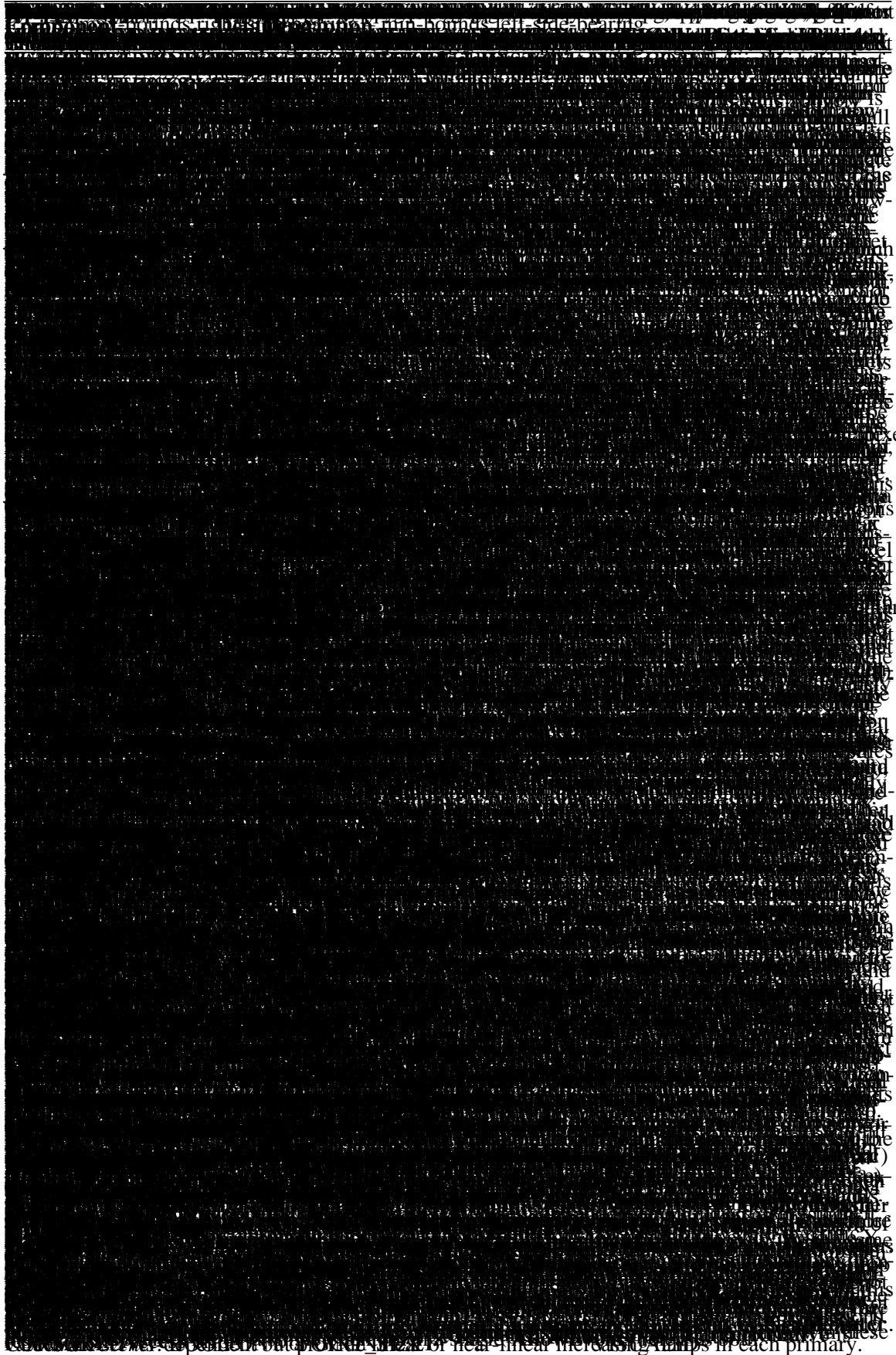
[REDACTED]

...near-linear increasing rumps in each primary.

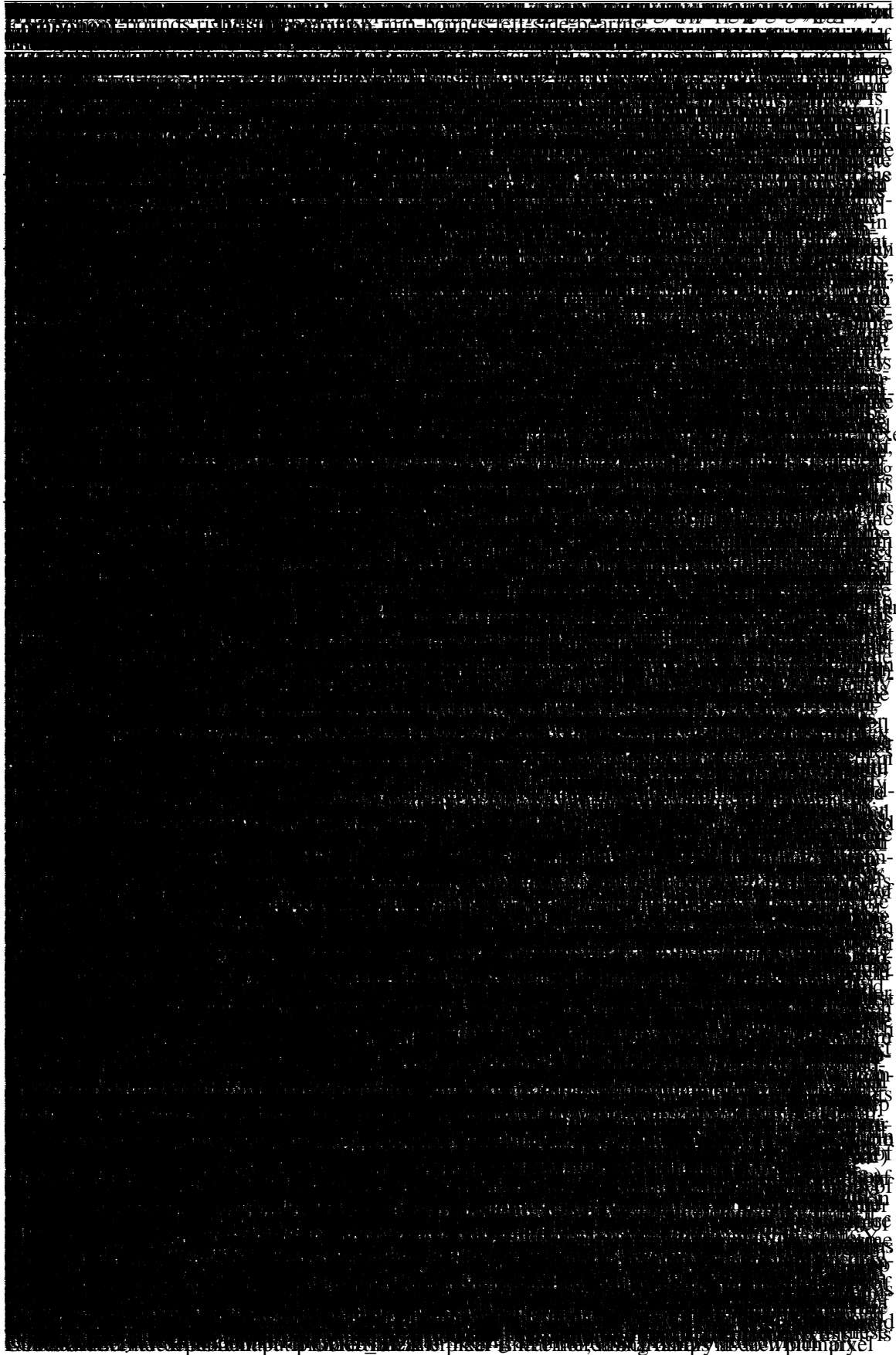
[REDACTED]

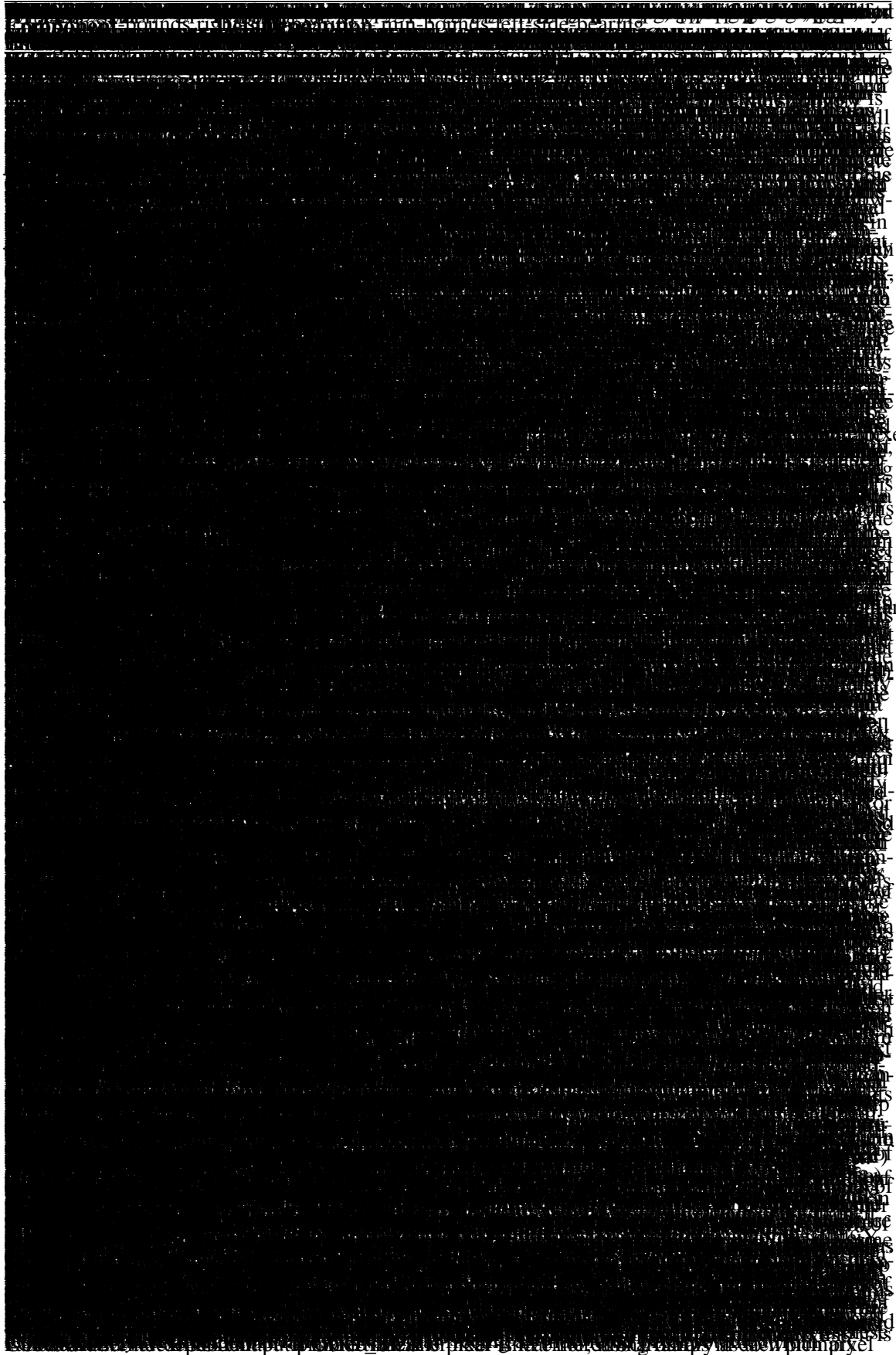
[REDACTED]

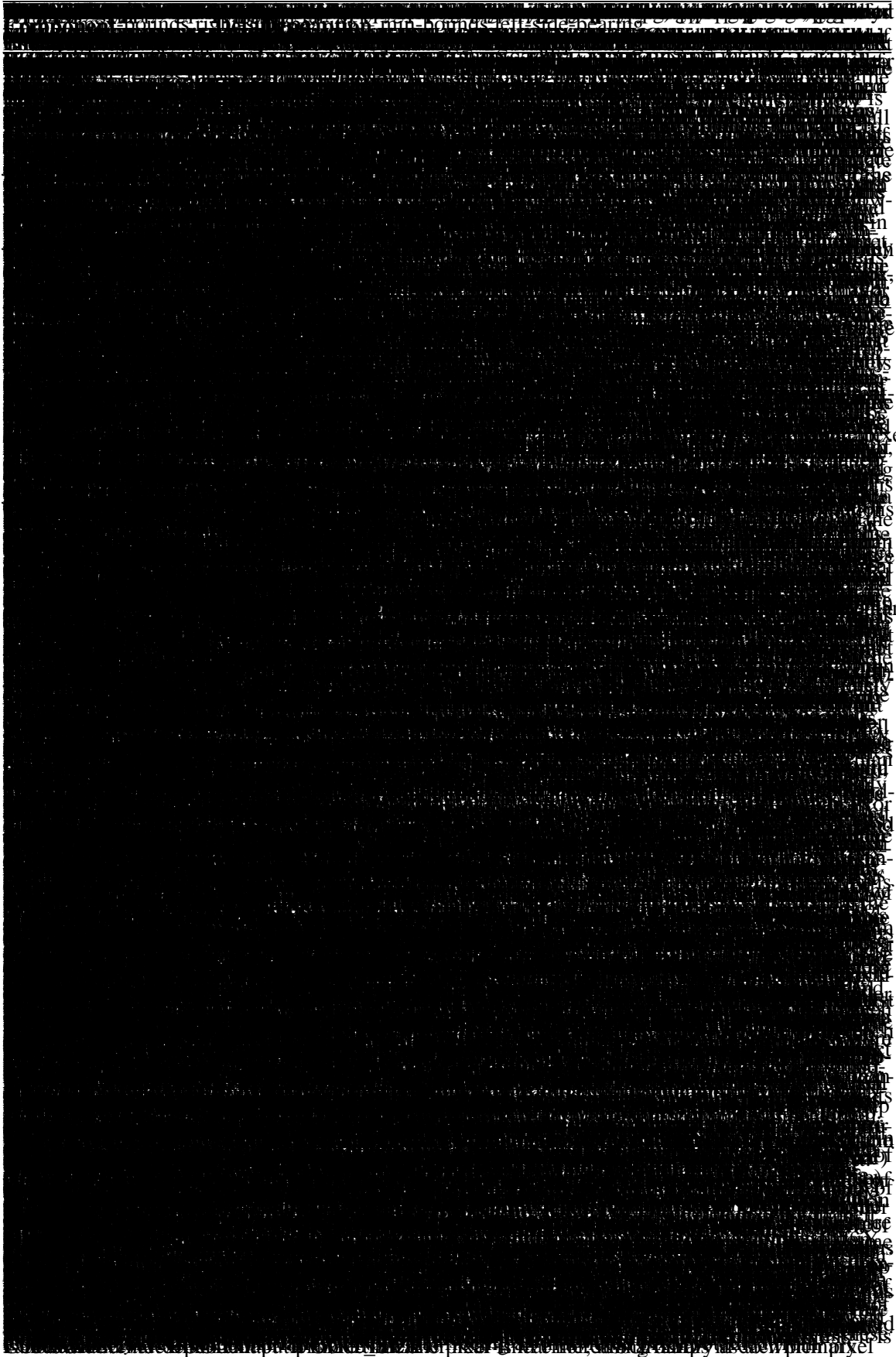
[REDACTED]

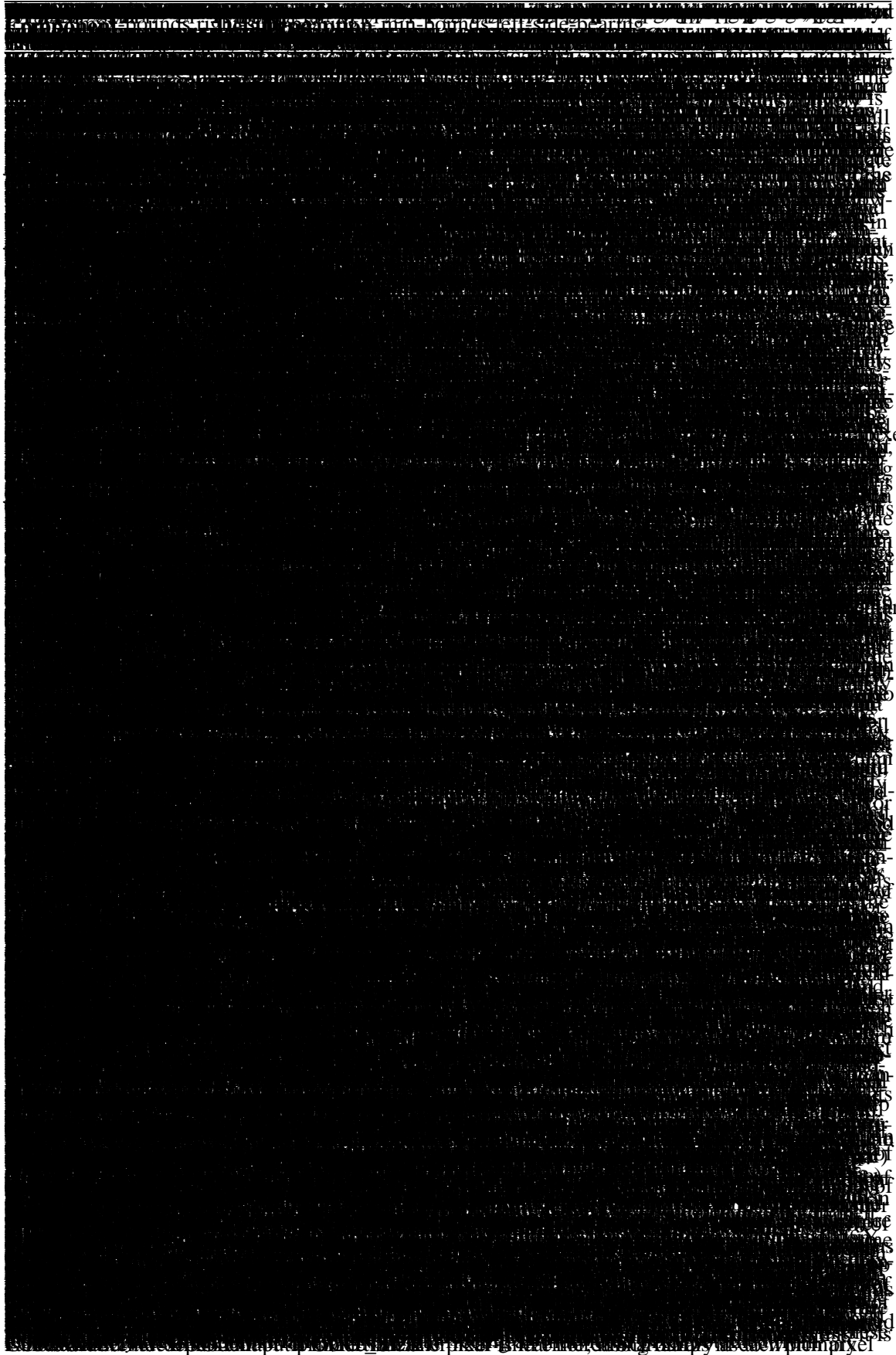


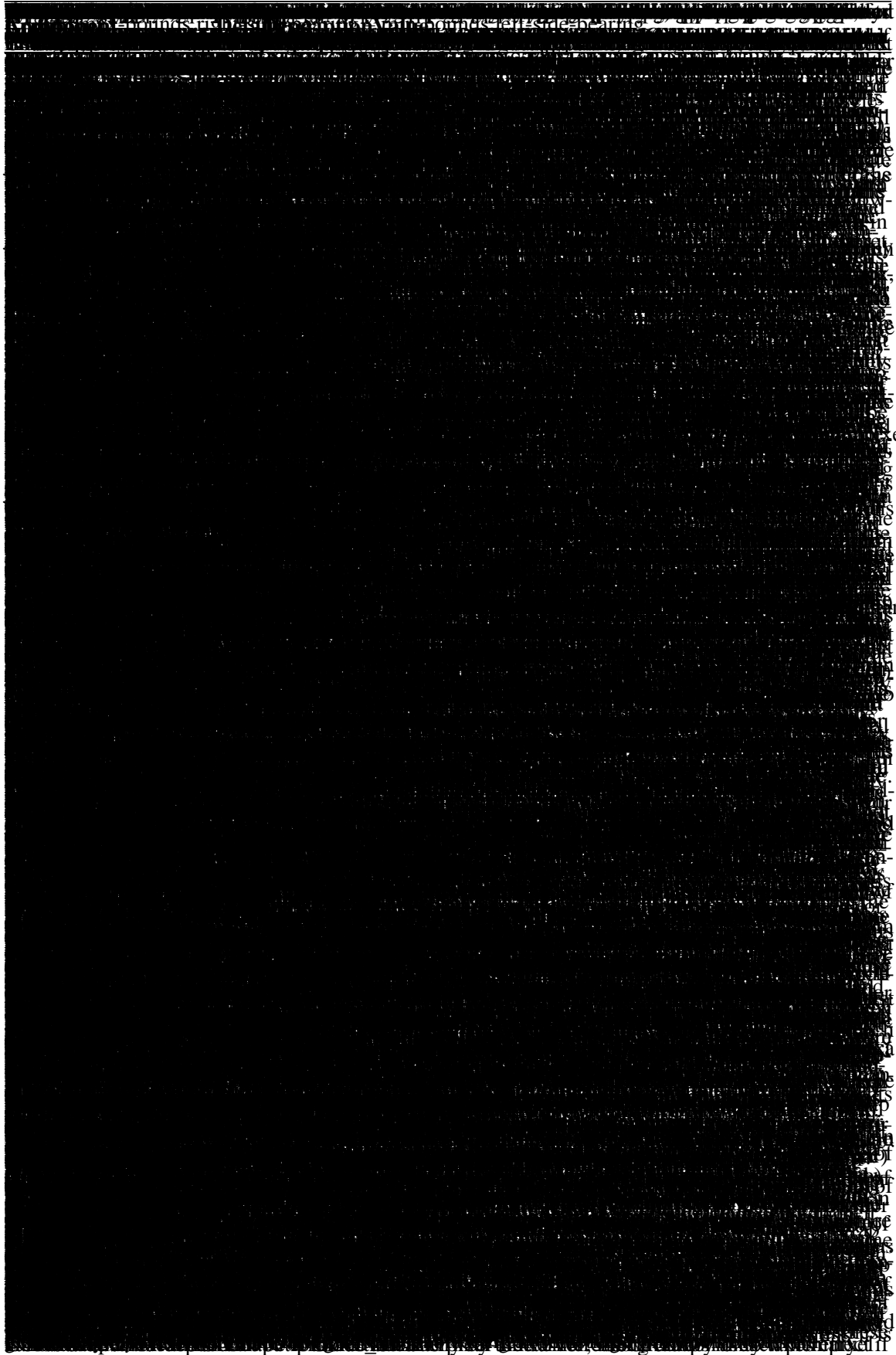
...was a period of only 10 min for near-linear increasing rumps in each primary.



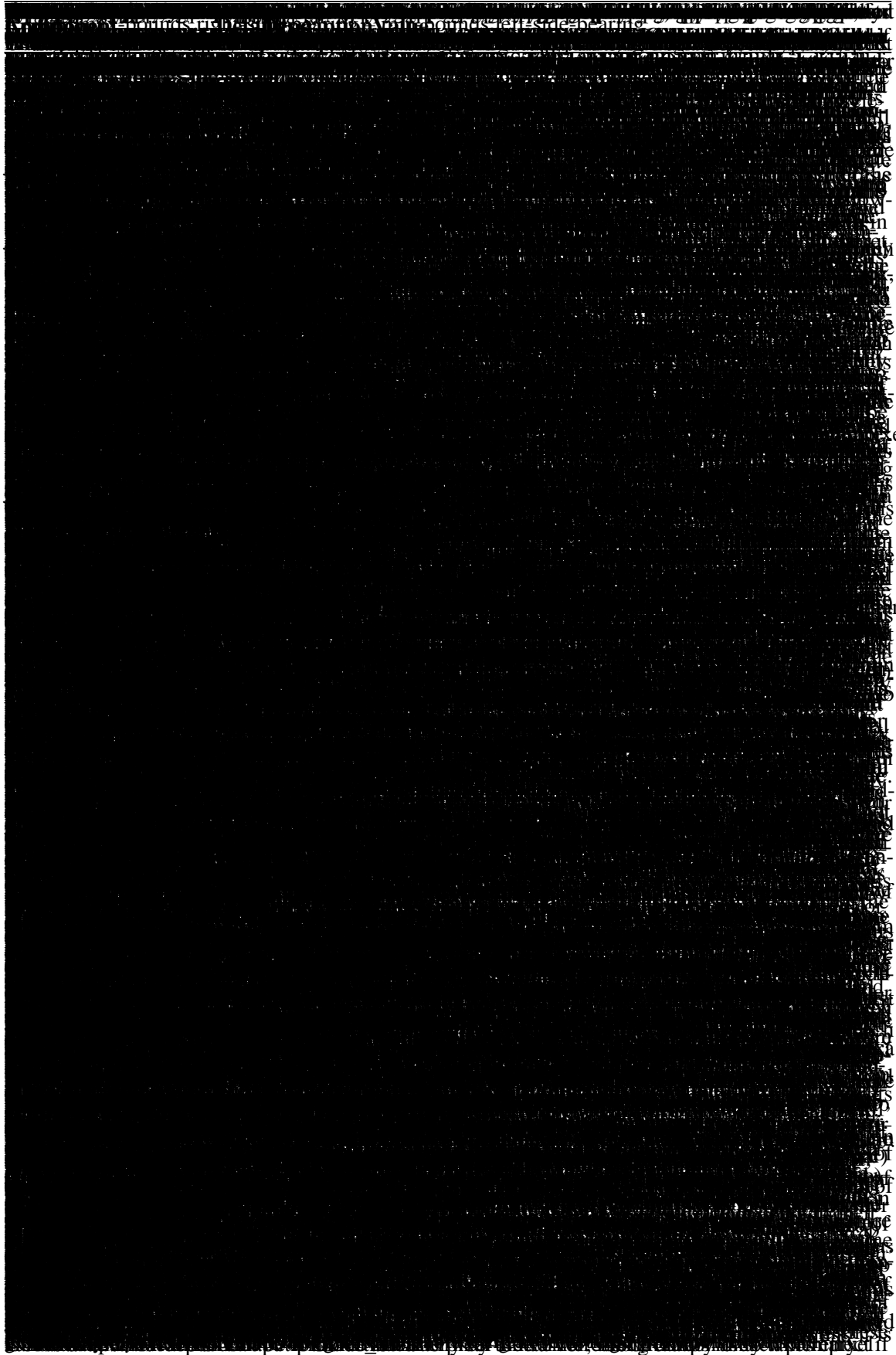




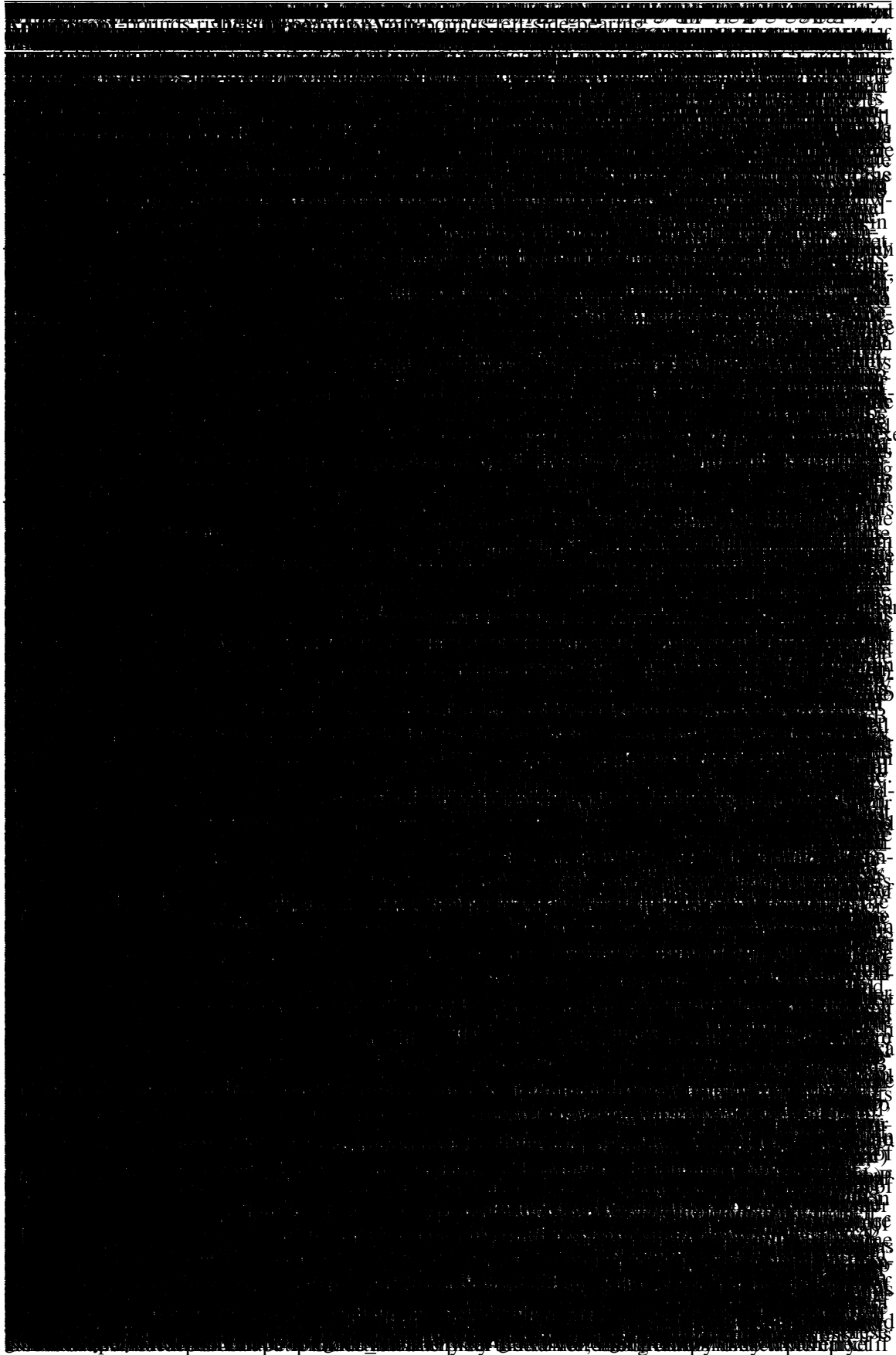




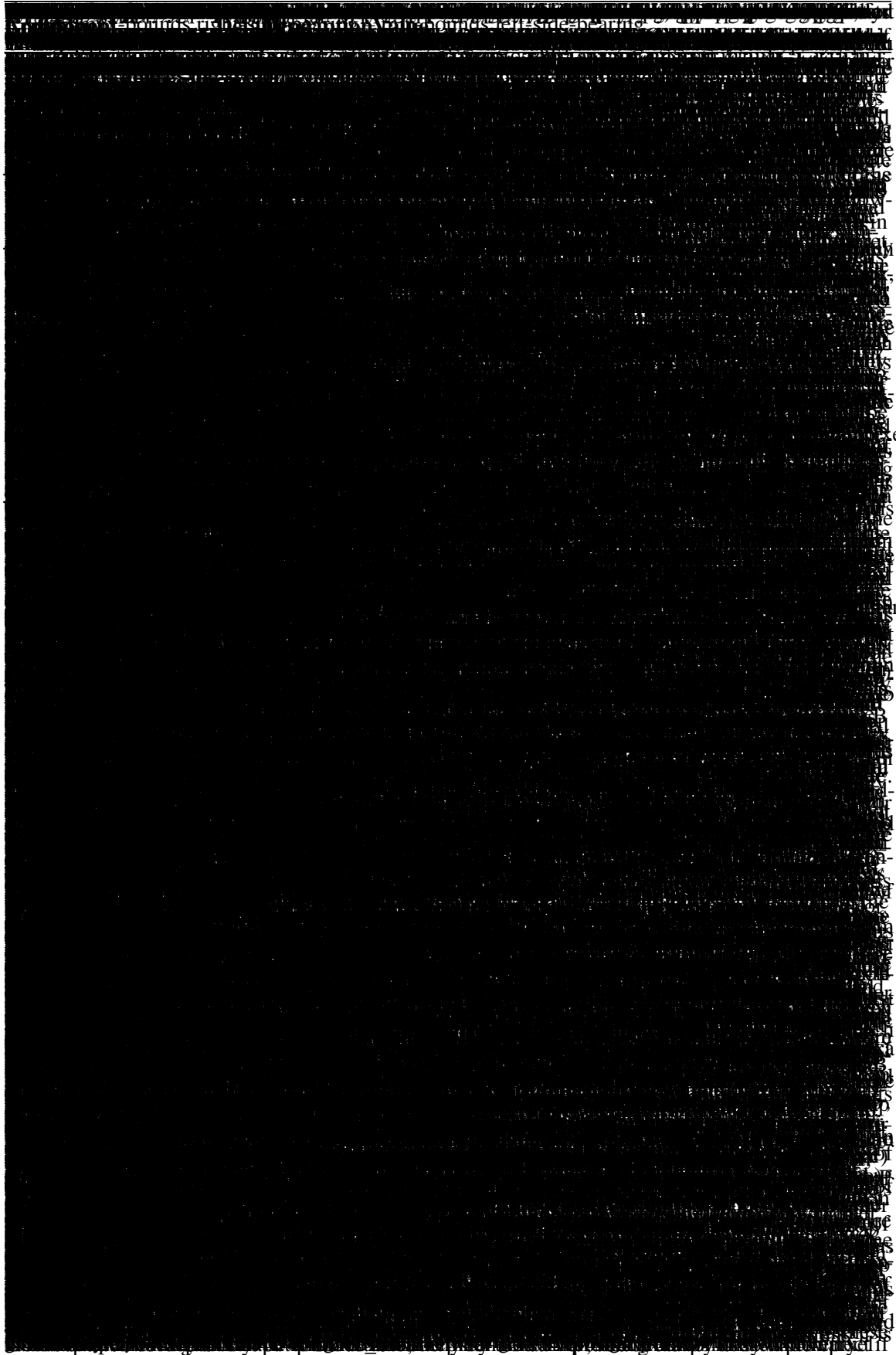
el
m-



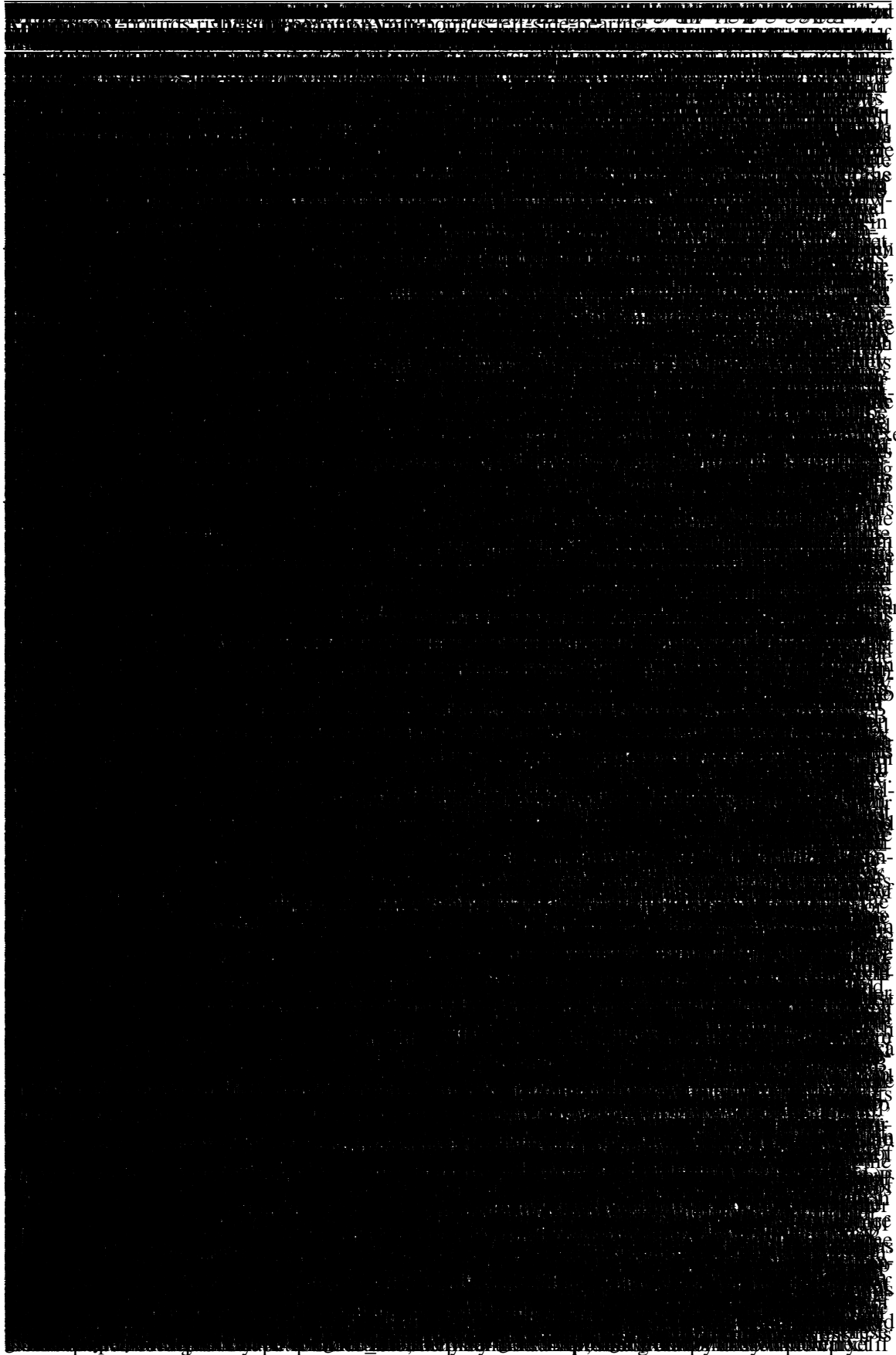
el
m-



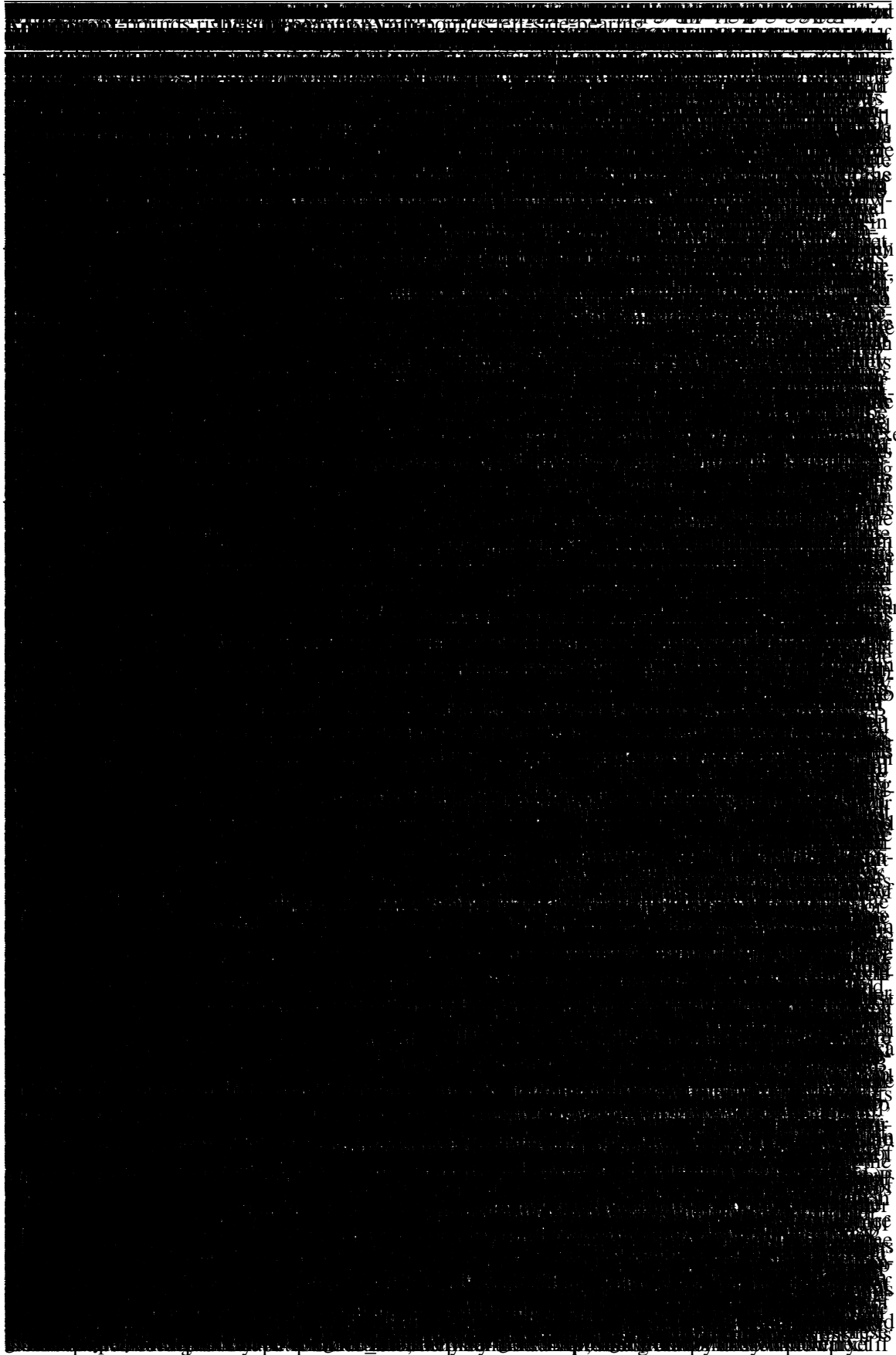
el
m-



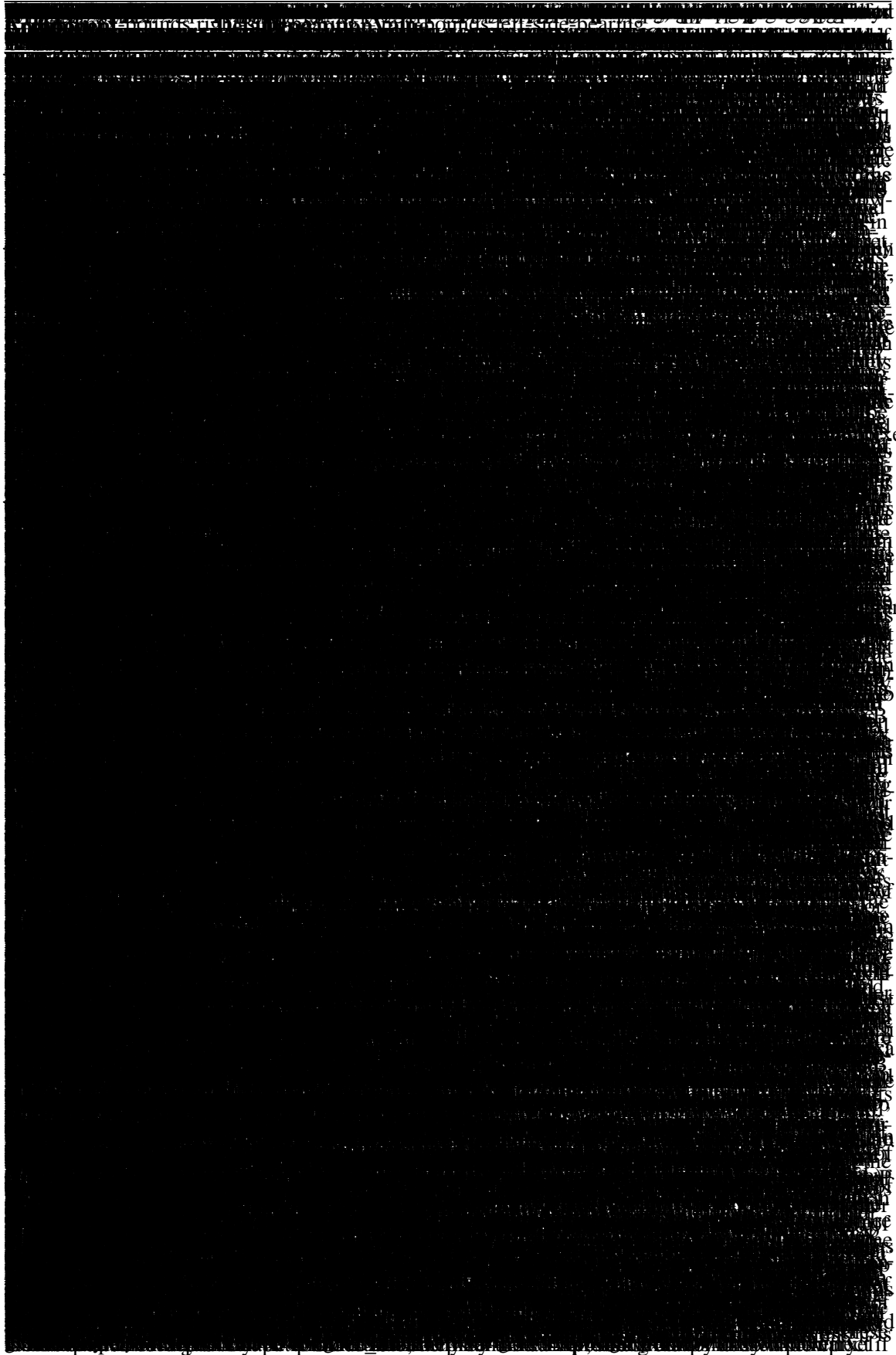
el
m-



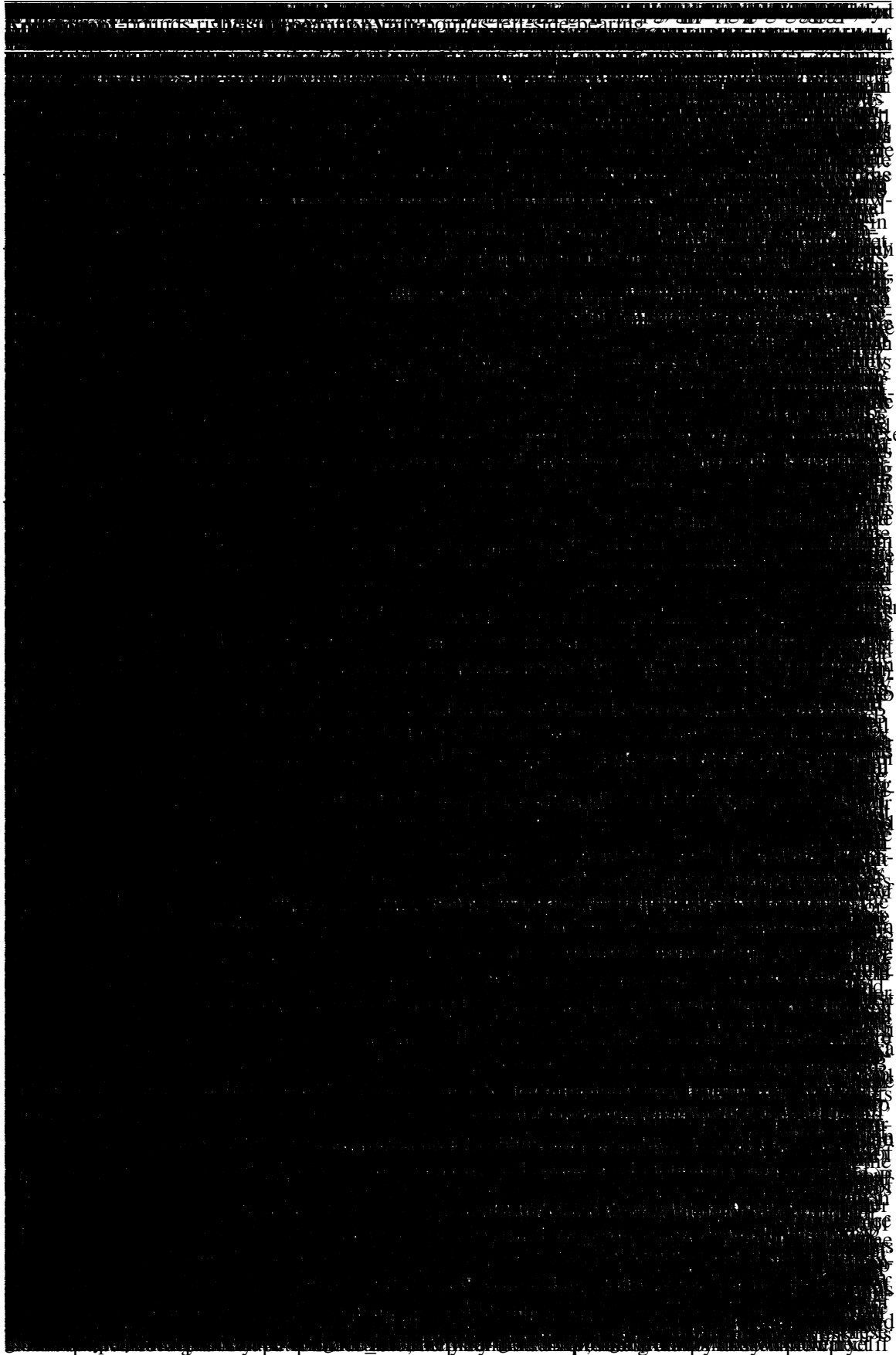
el
m-



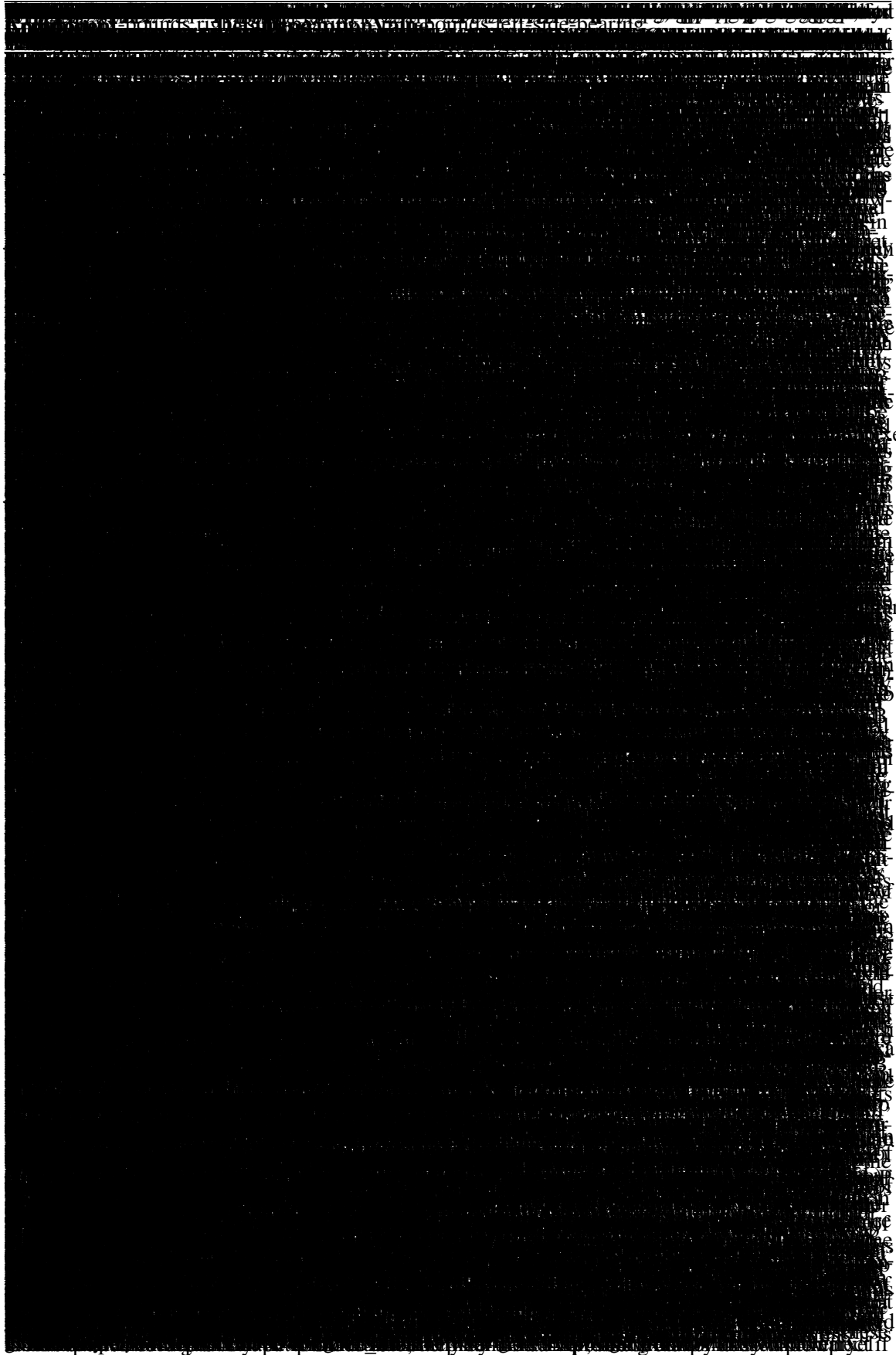
el
m-



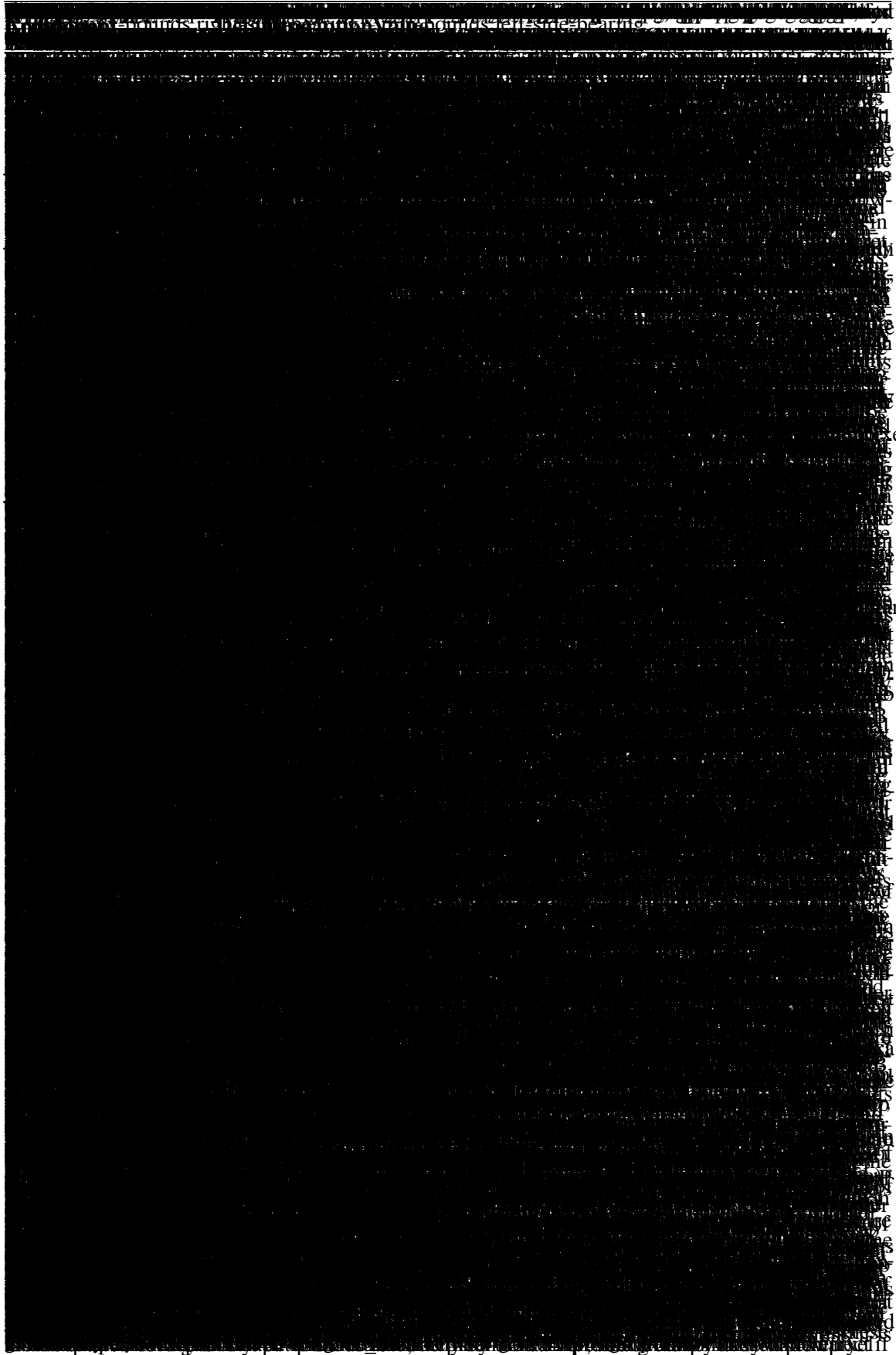
el
m-



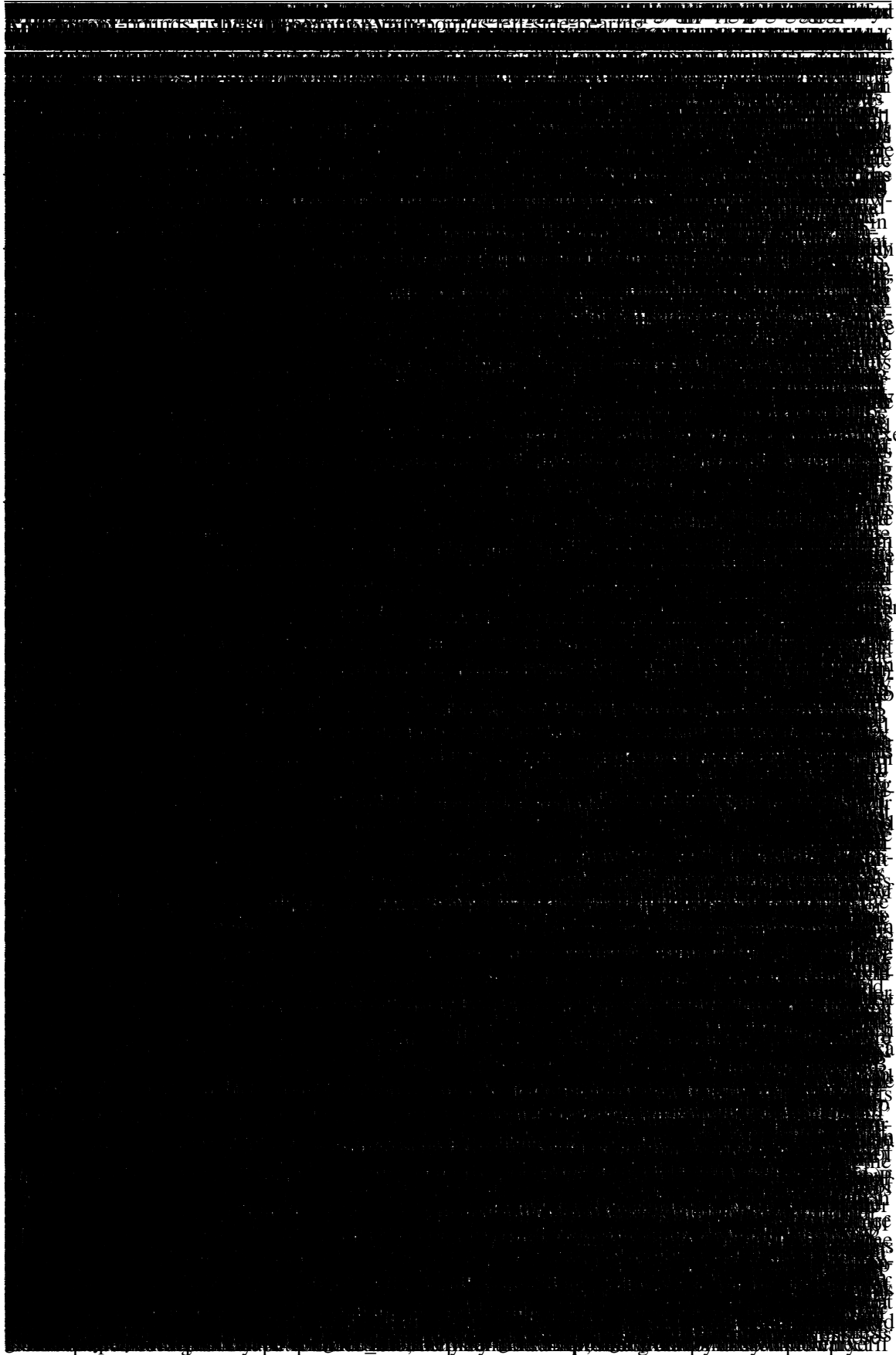
el
m-



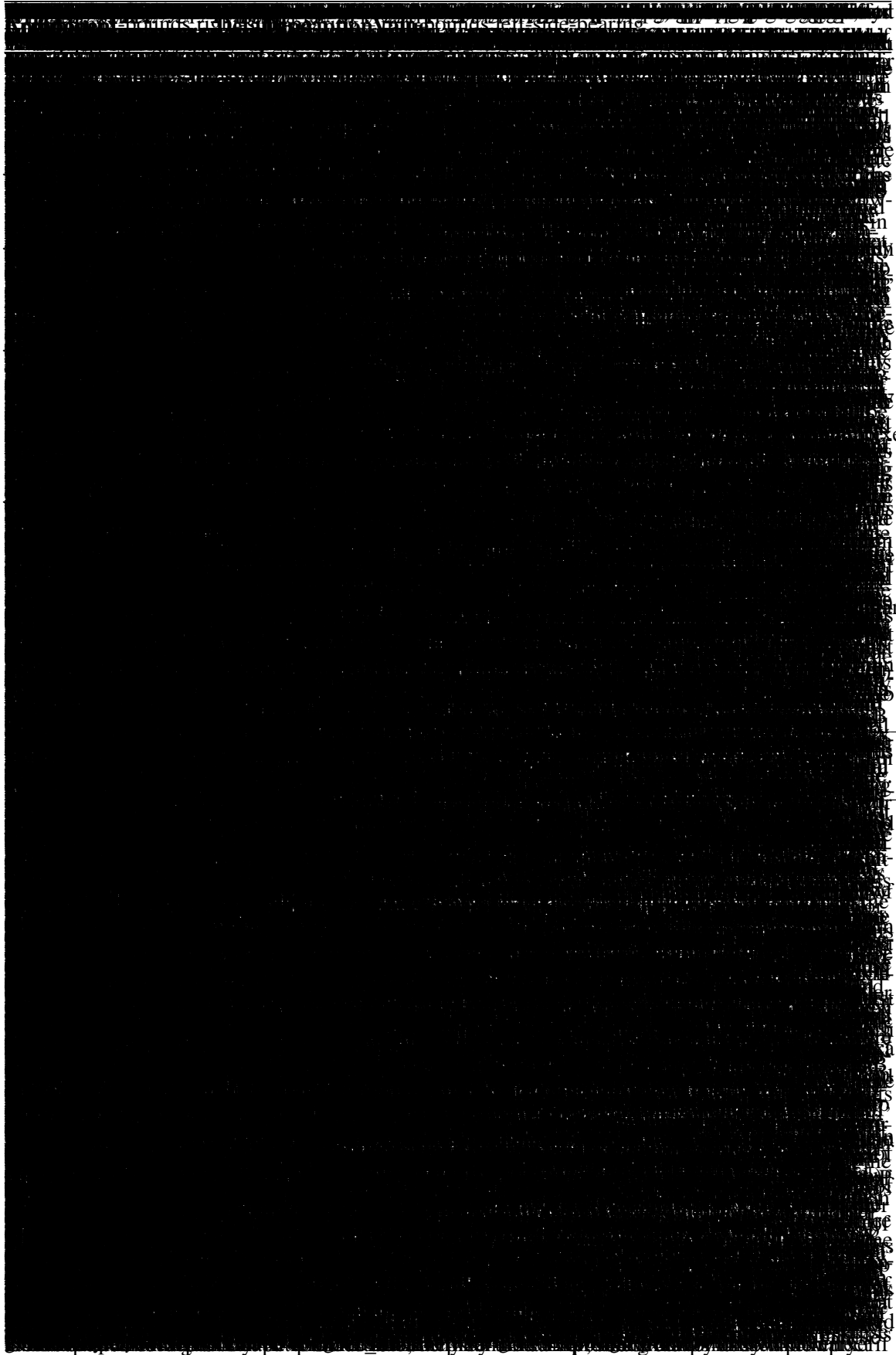
el
m-



el
m-



el
m-



el

m-

[REDACTED]

el
m-

[REDACTED]

el
m-

[REDACTED]

el
m-

[REDACTED]

el
m-

[REDACTED]

el
m-

[REDACTED]

el
m-

[REDACTED]

el
m-

[REDACTED]

el
m-

[REDACTED]

el
m-

[REDACTED]

el
am-

[REDACTED]

el
am-

[REDACTED]

el
am-

[REDACTED]

el
m-

[REDACTED]

el

m-

[REDACTED]

el

m-

[REDACTED]

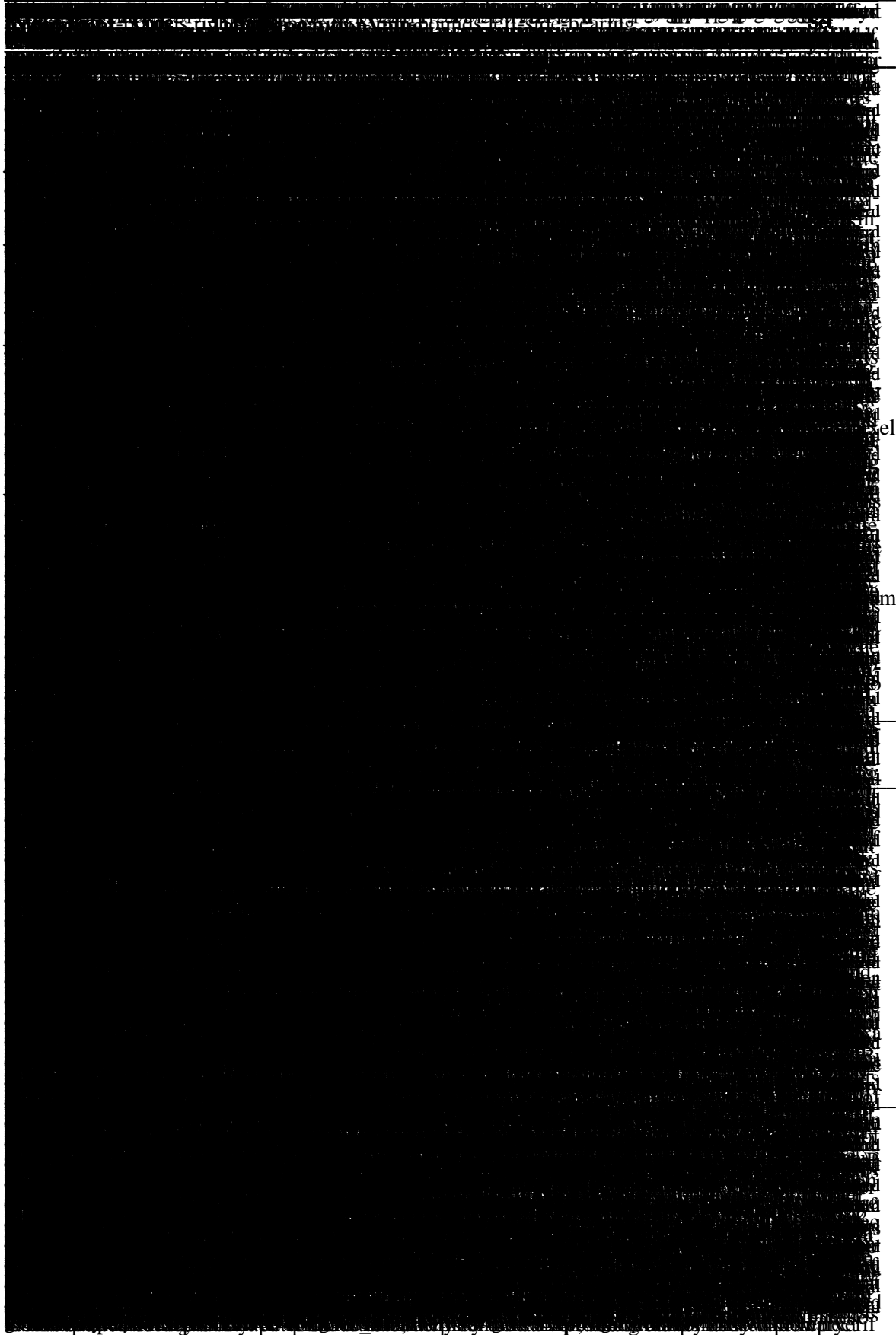
el

m-

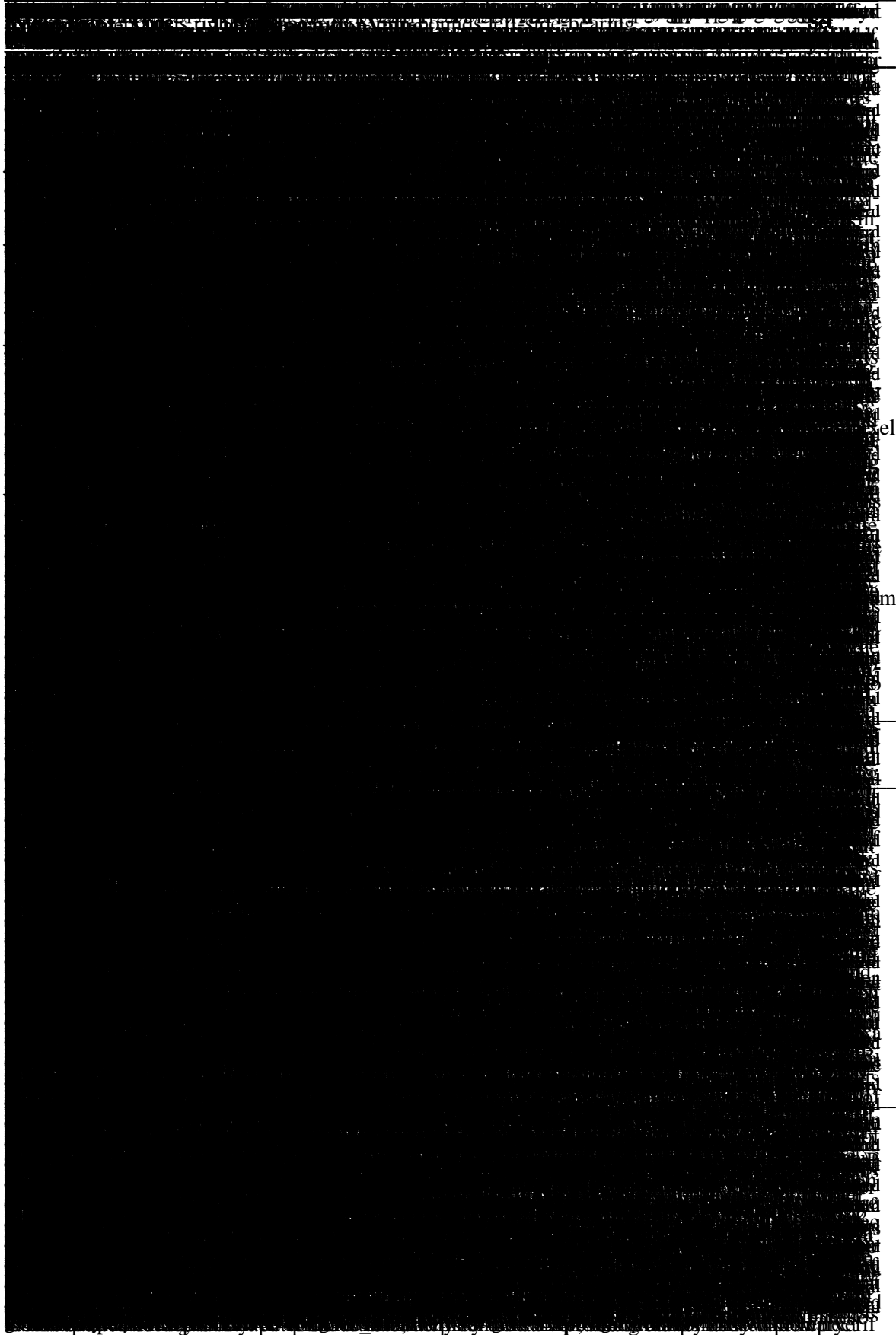
[REDACTED]

el

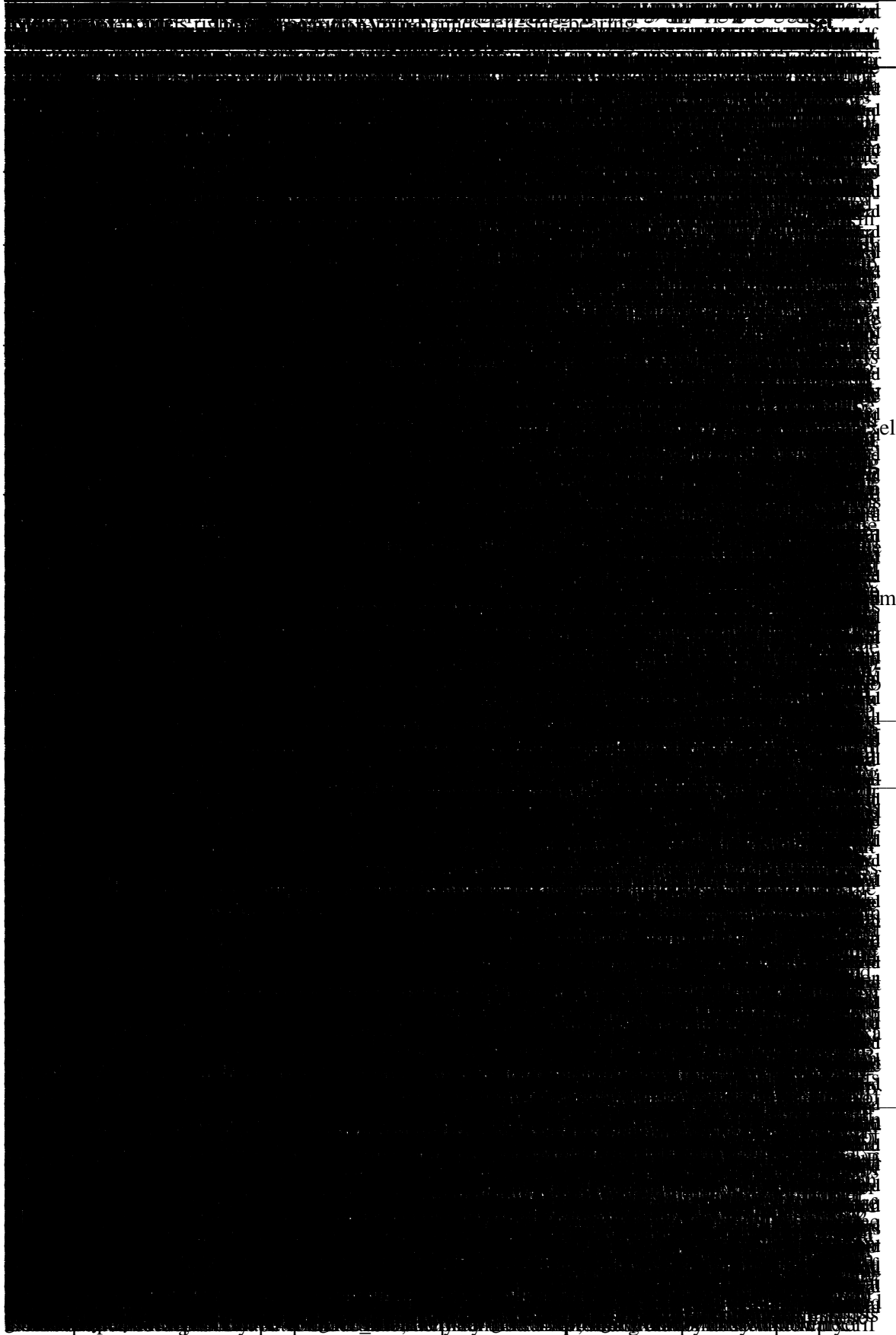
m-



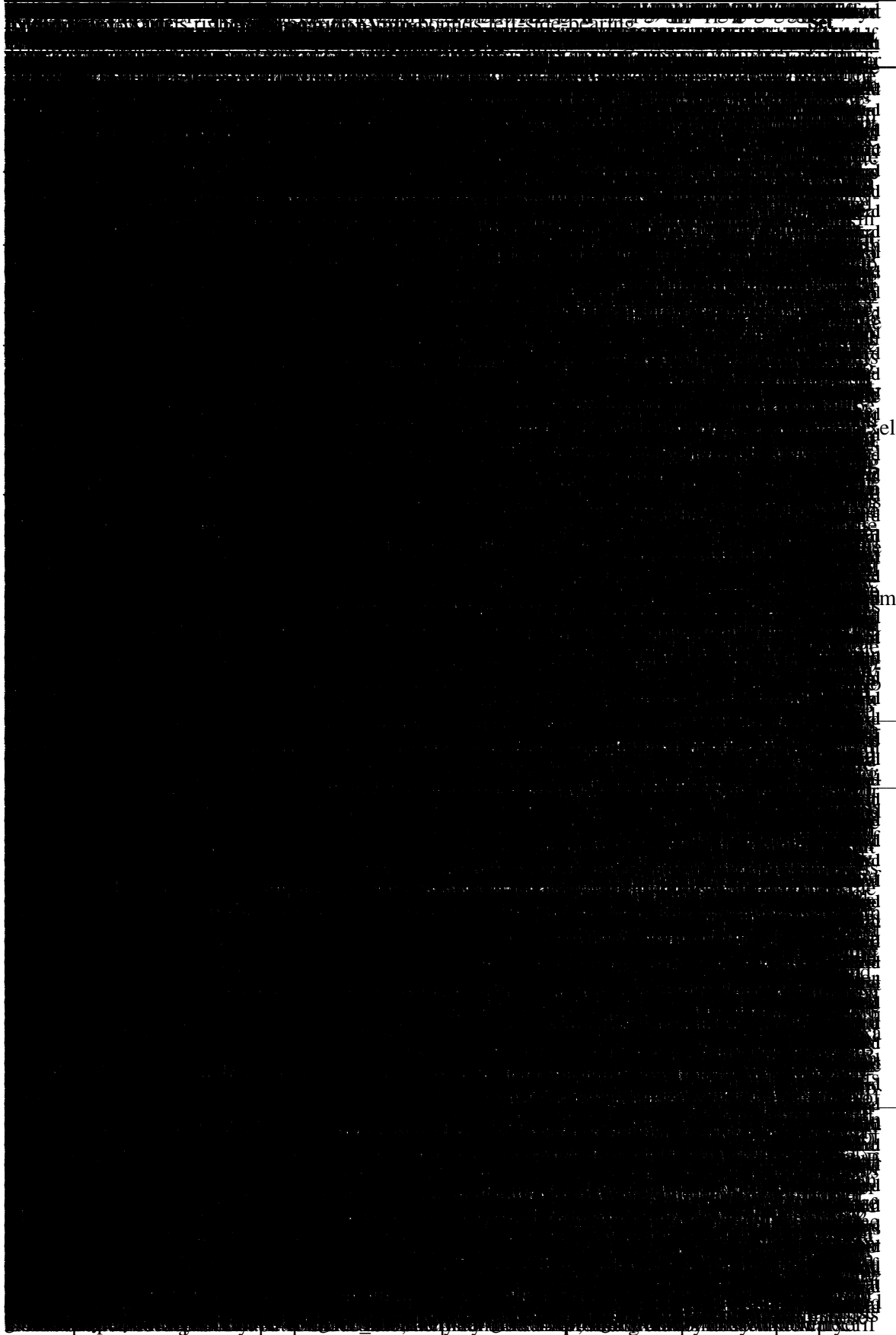
el
m-



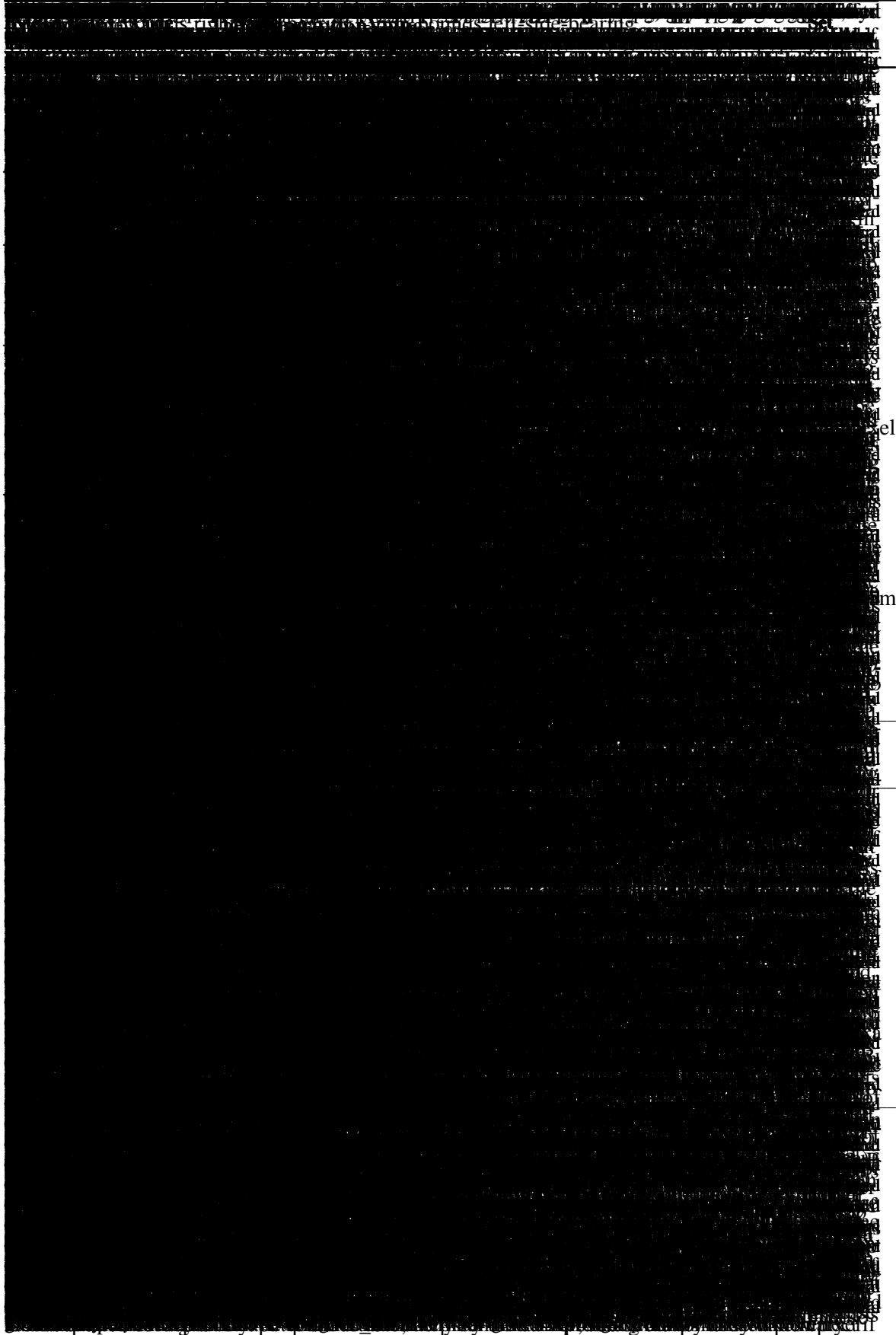
el
m-



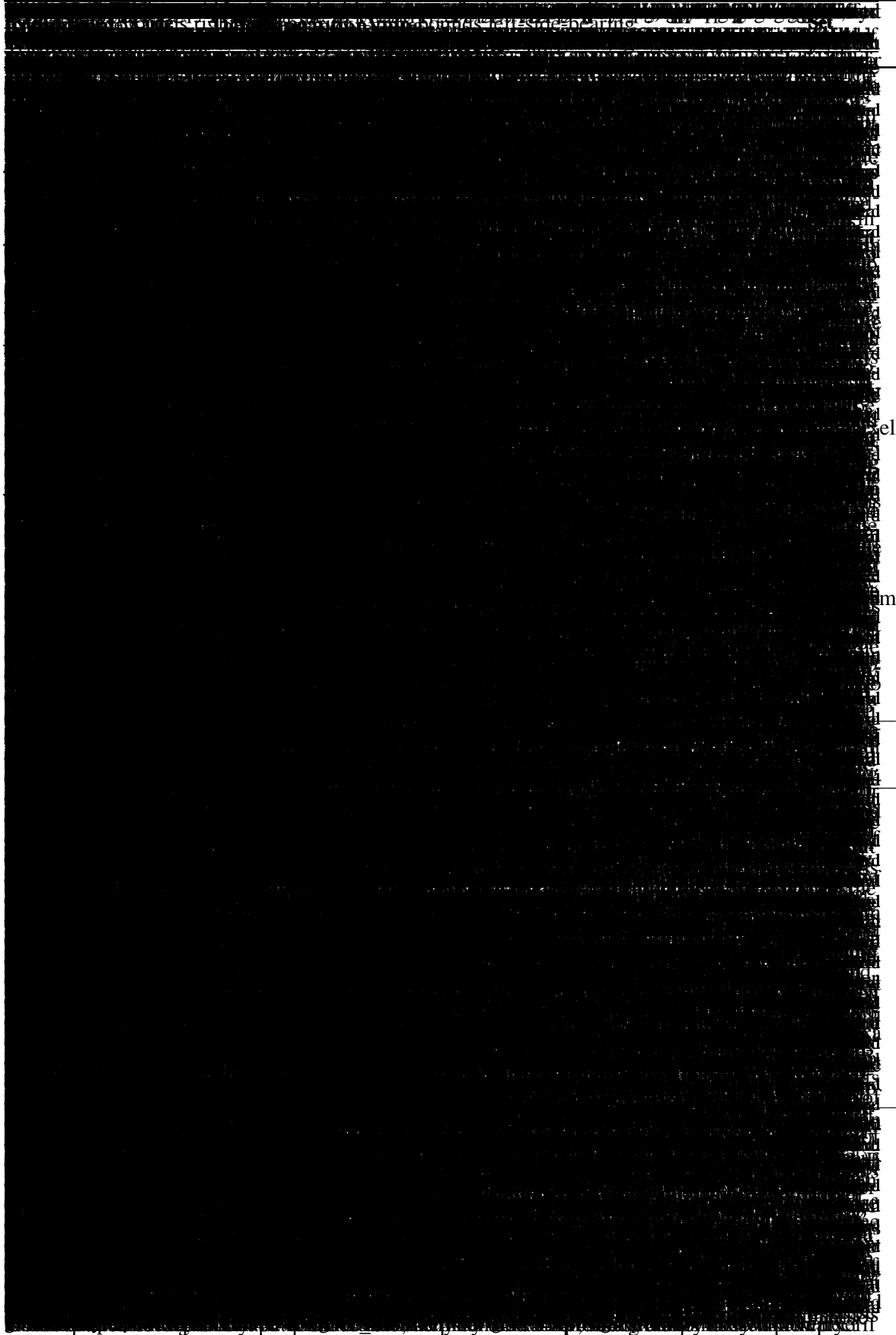
el
m-



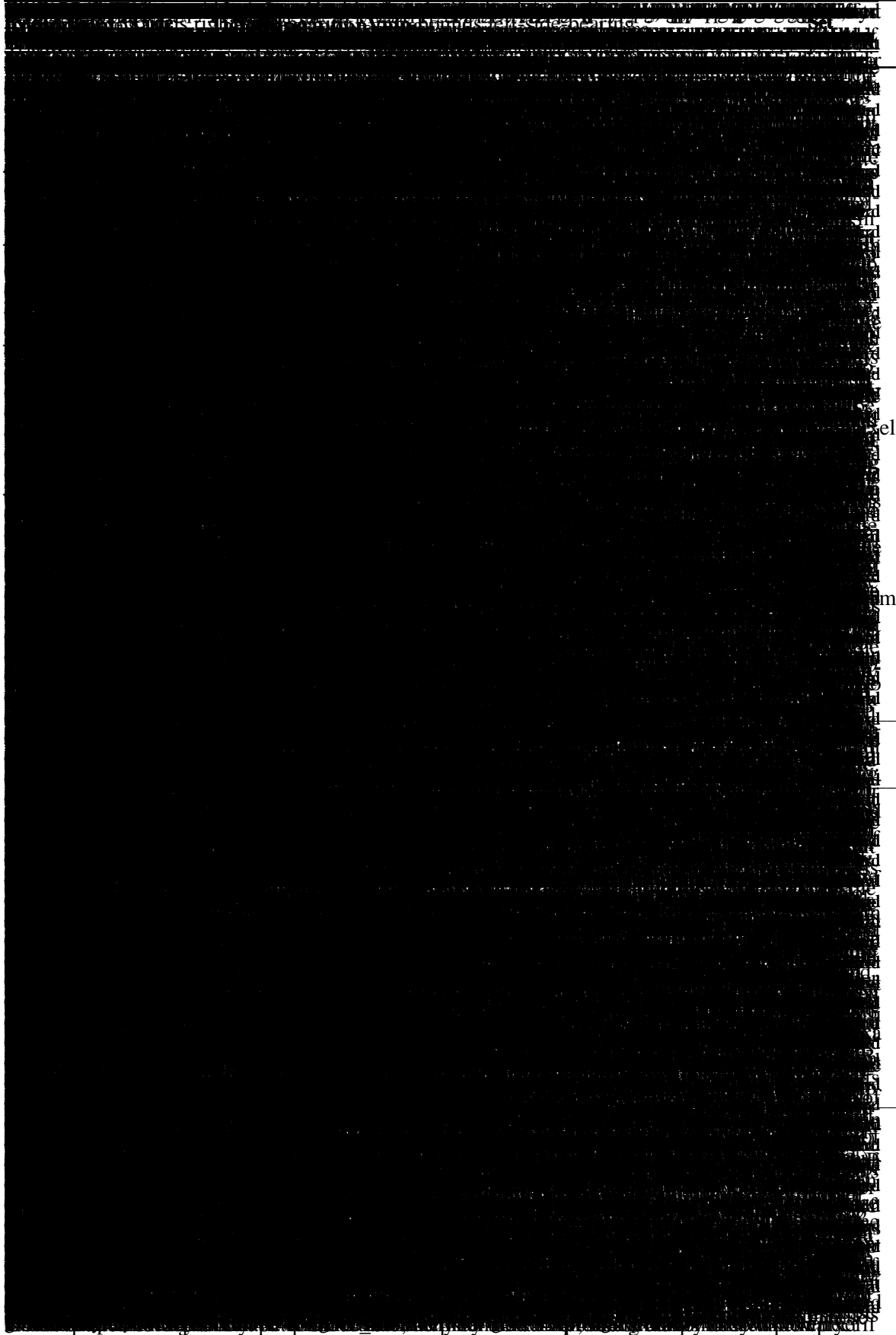
el
m-



el
m-

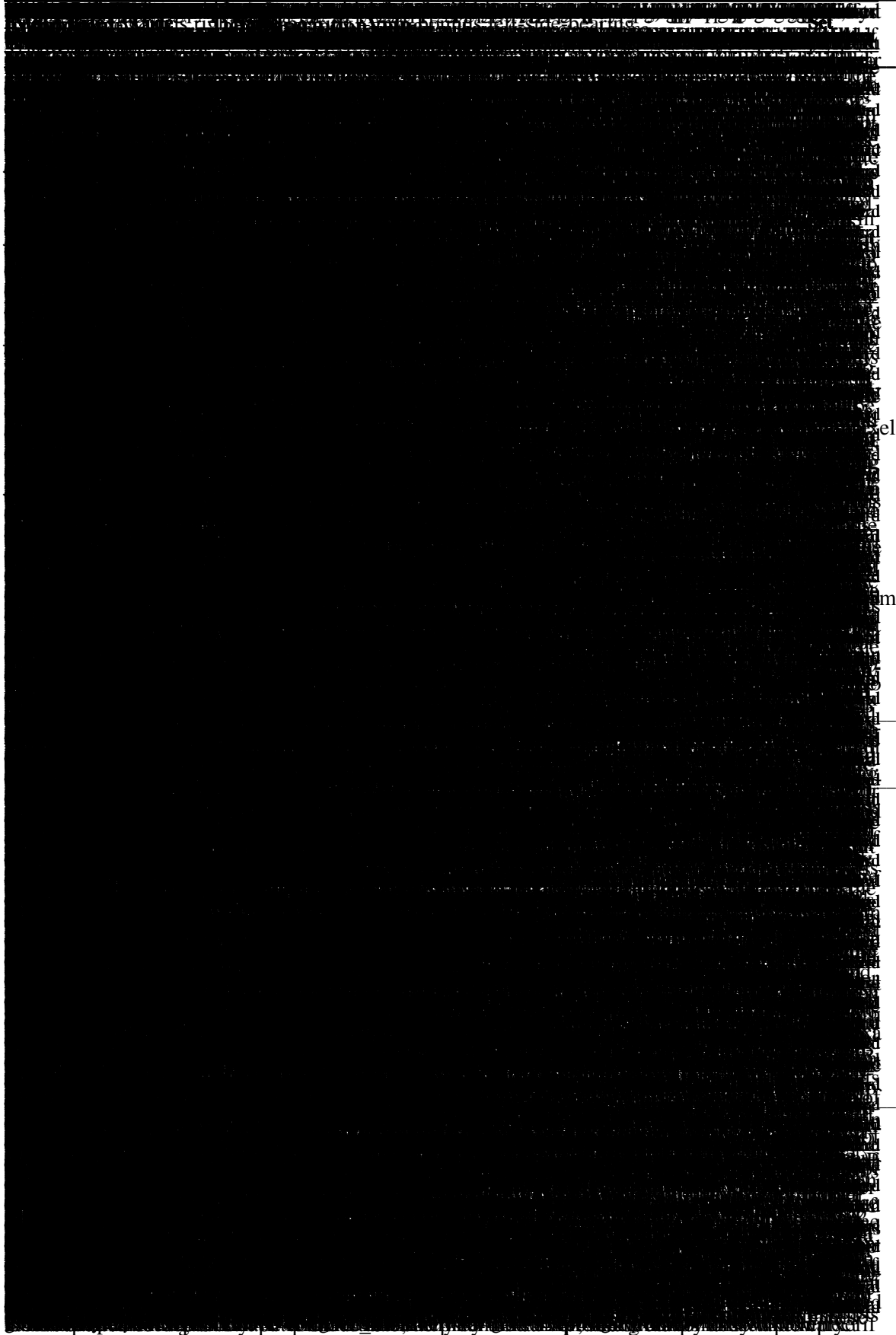


el
m-



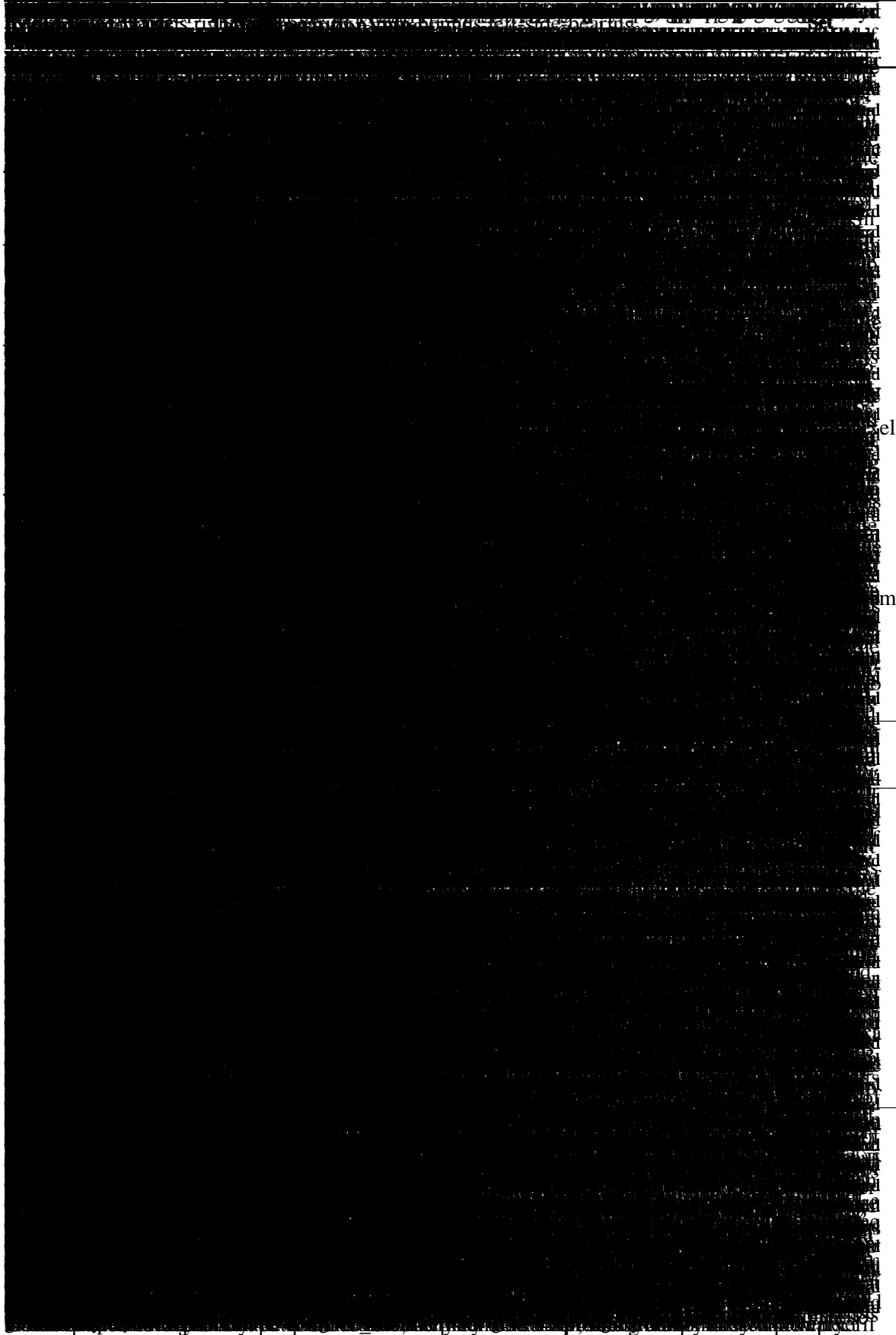
el

m-

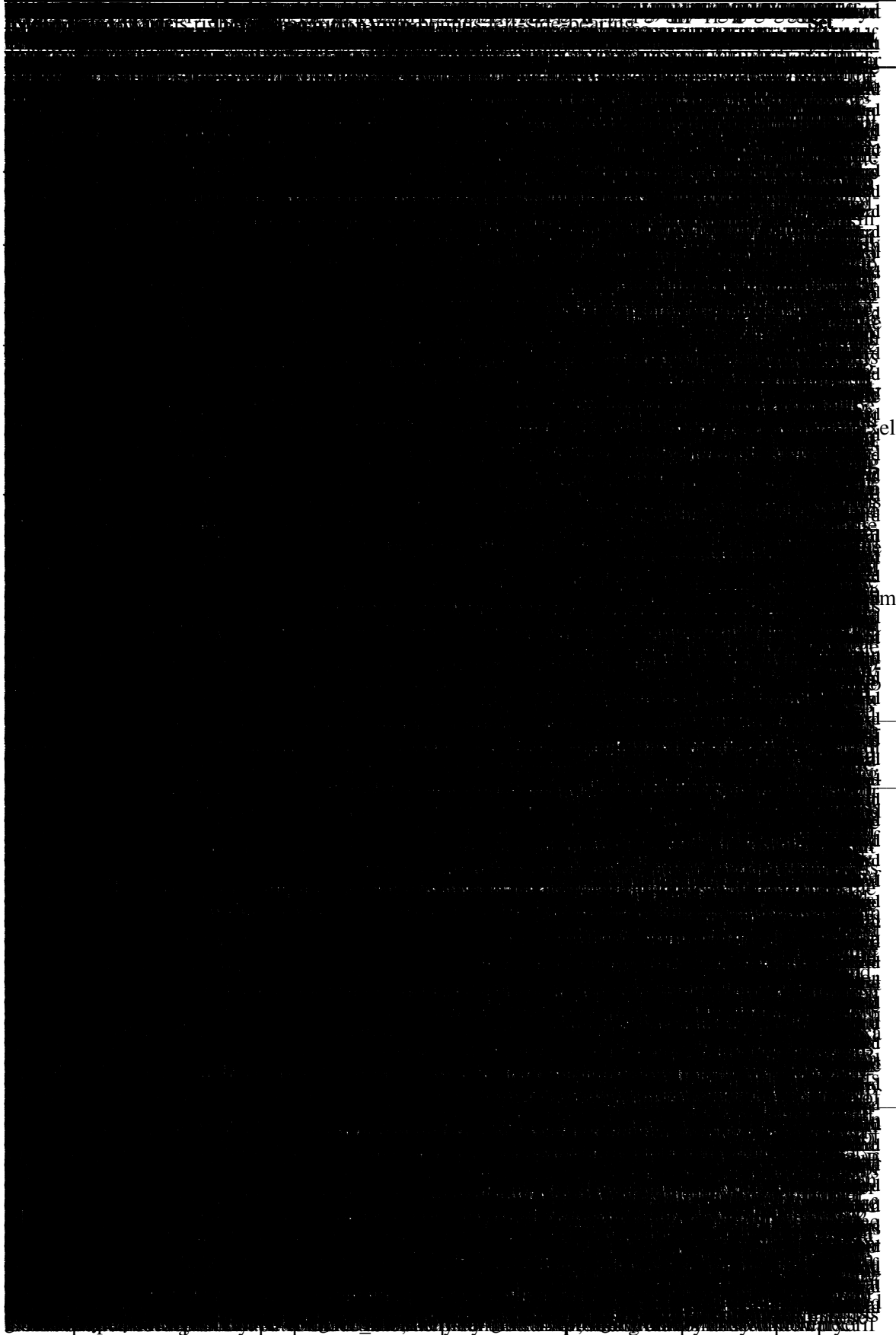


el

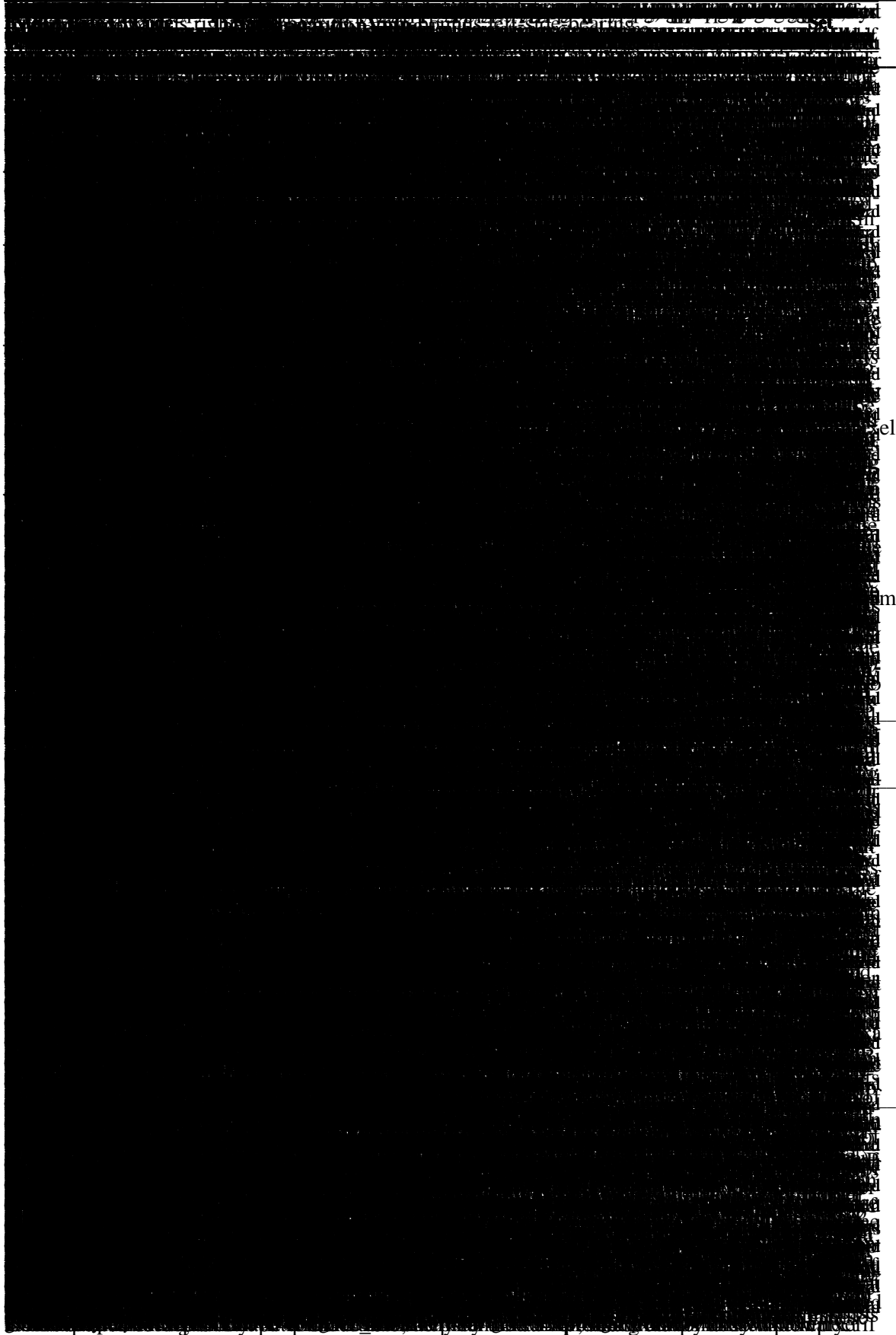
m-



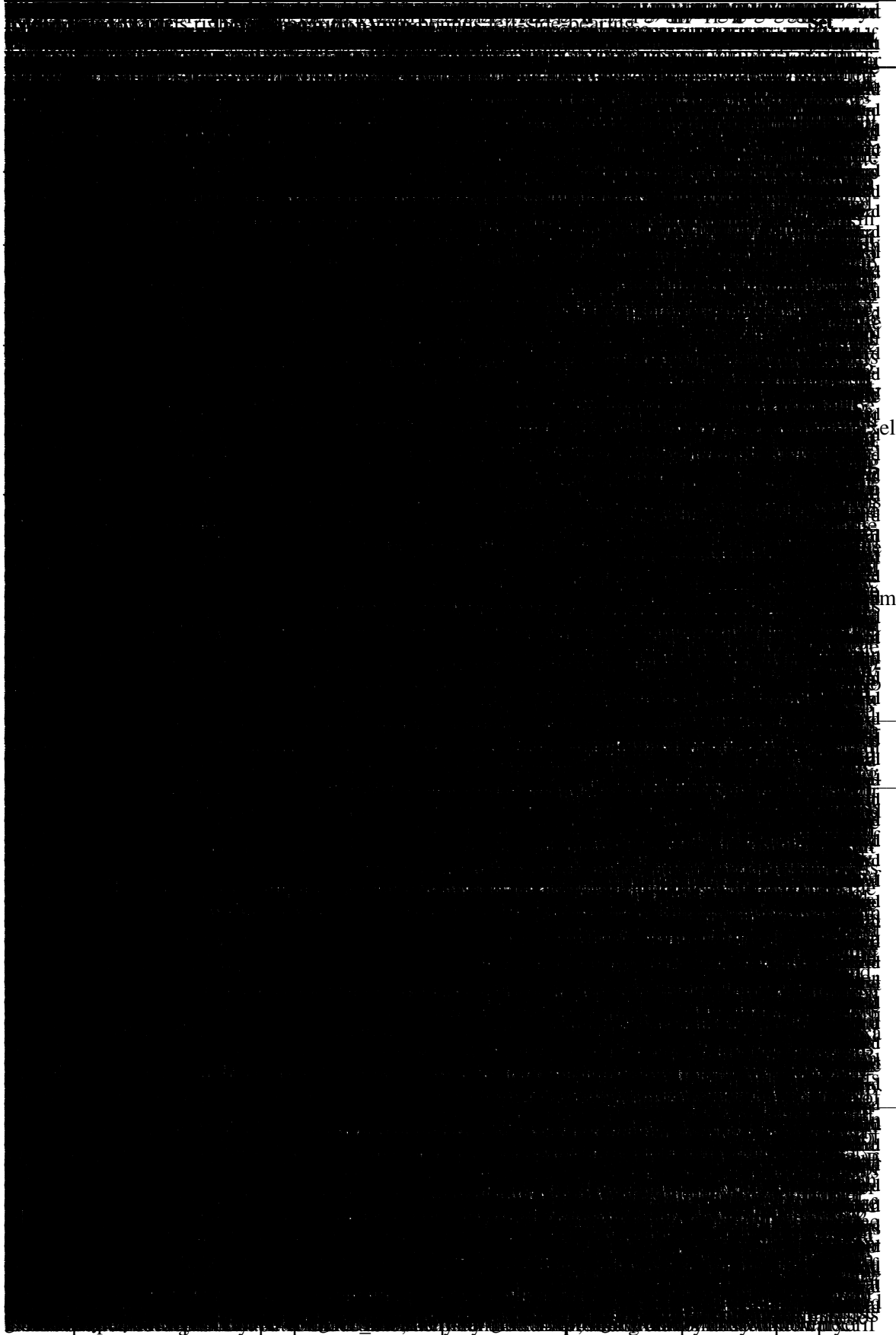
el
m-



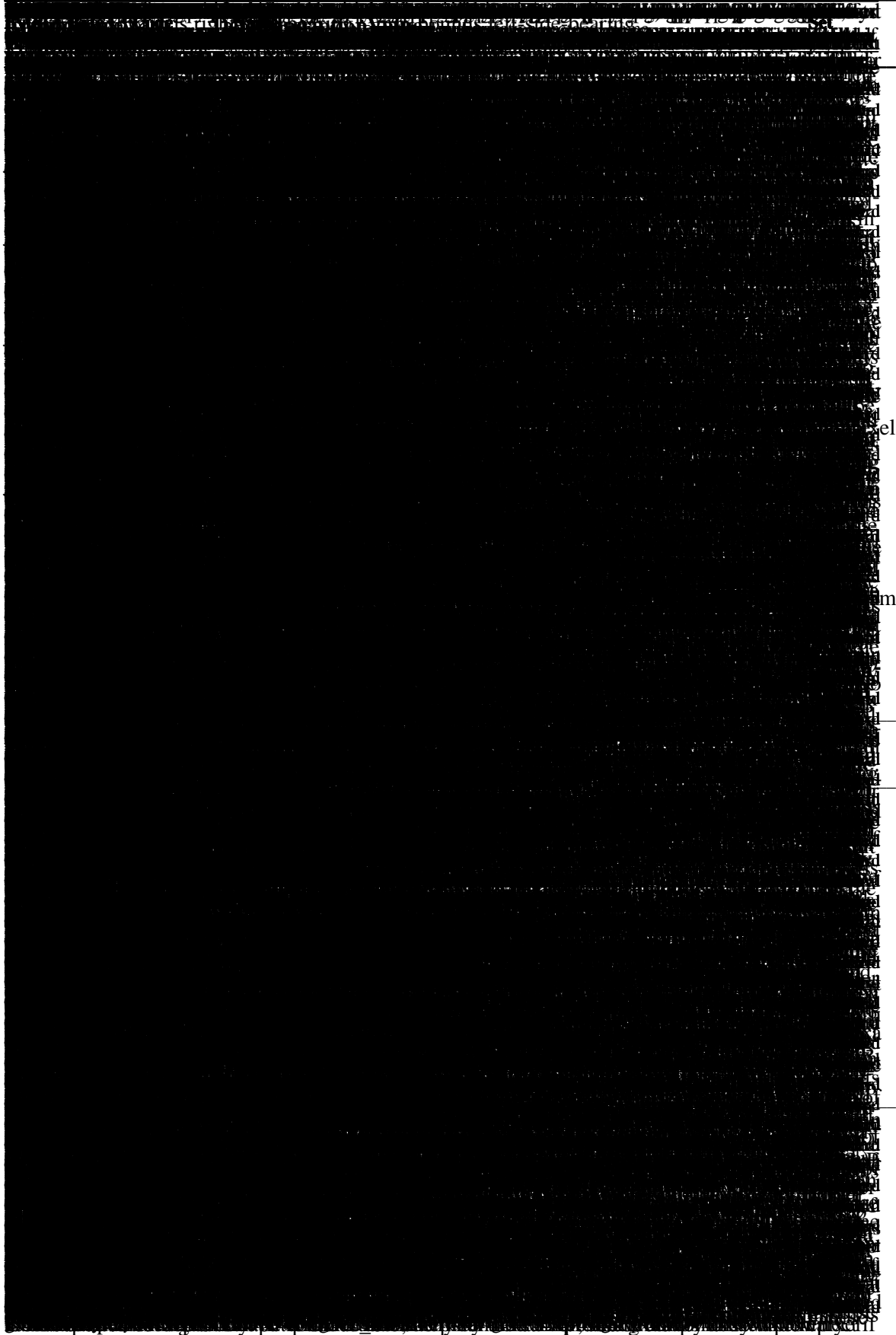
el
m-



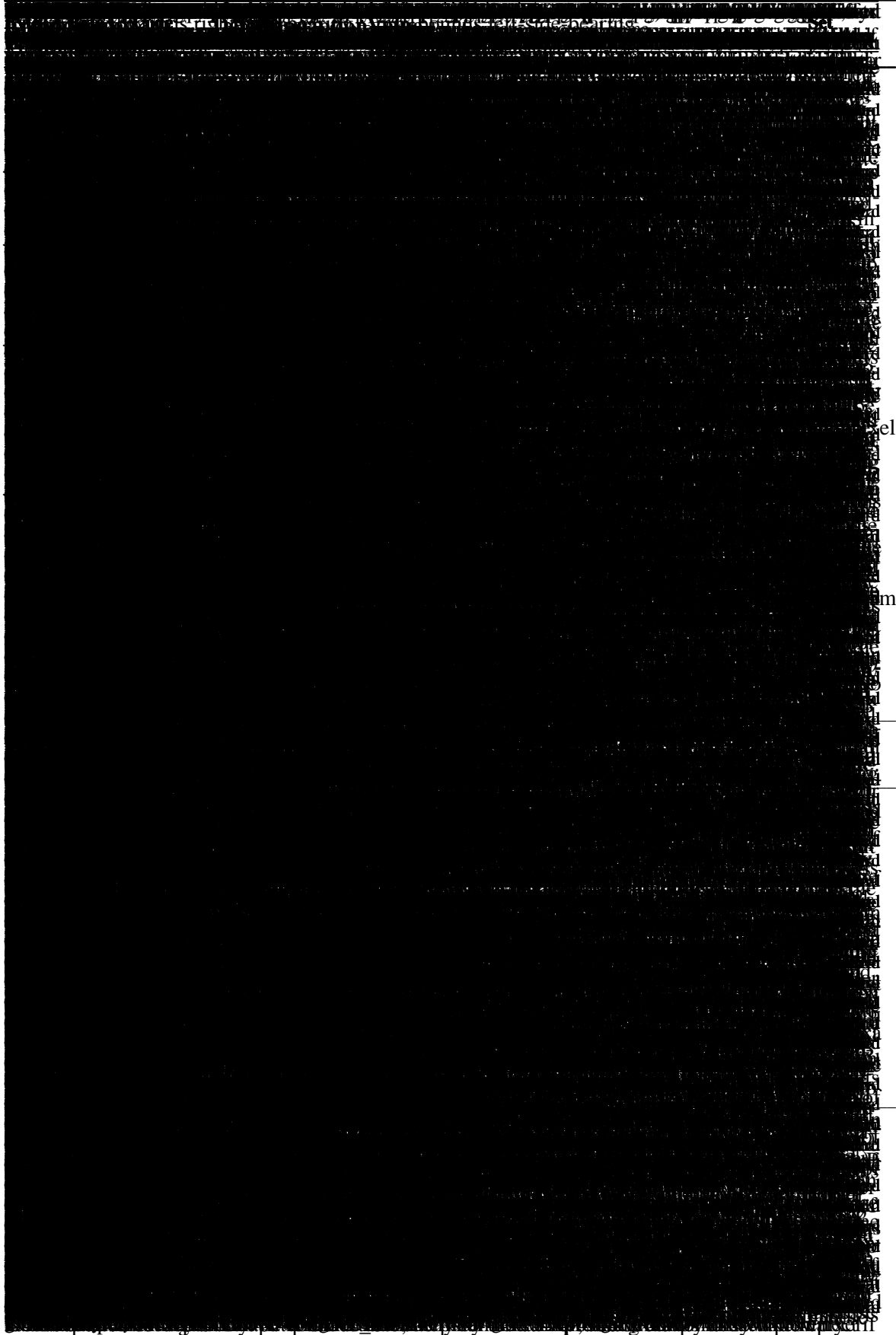
el
m-



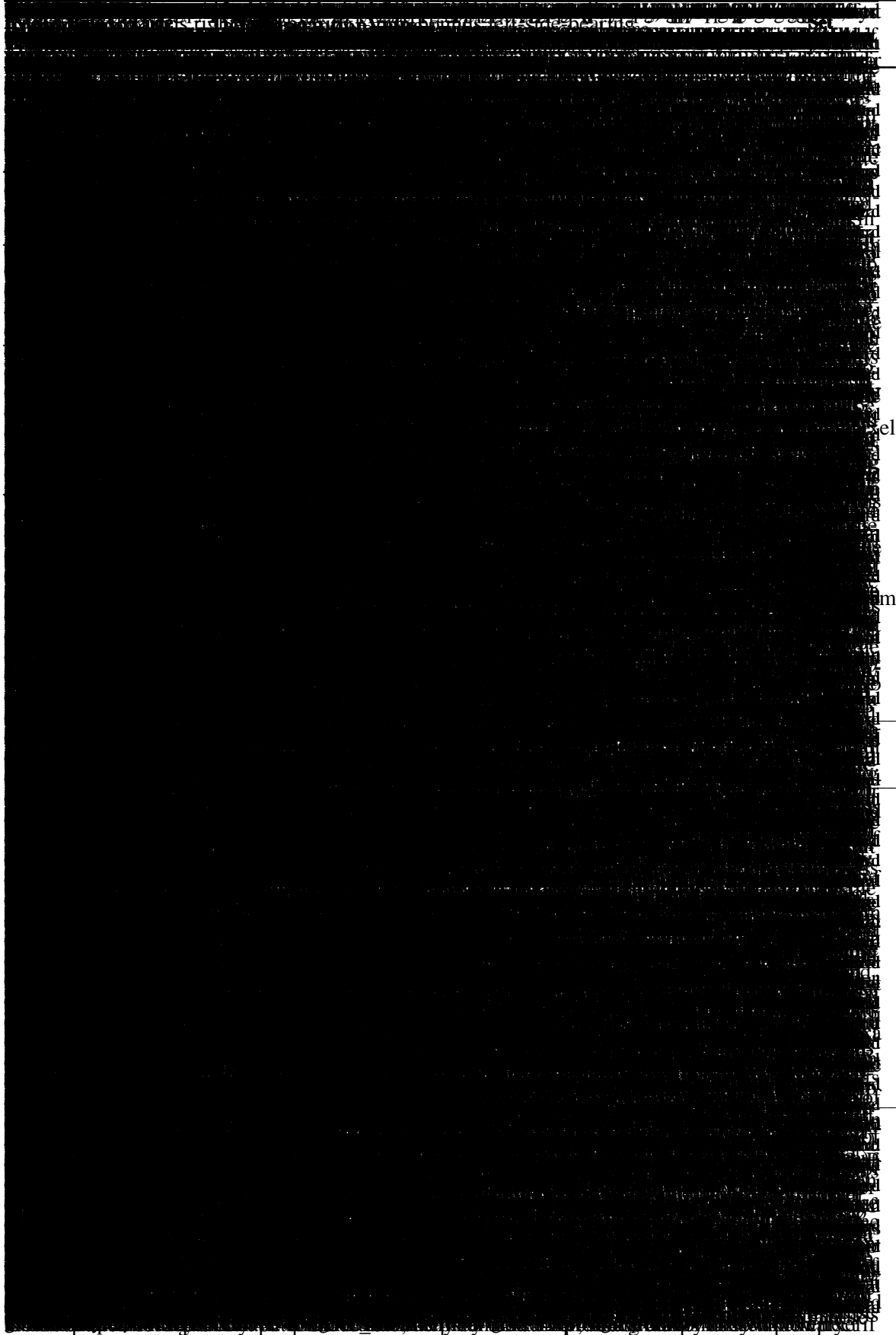
el
m-



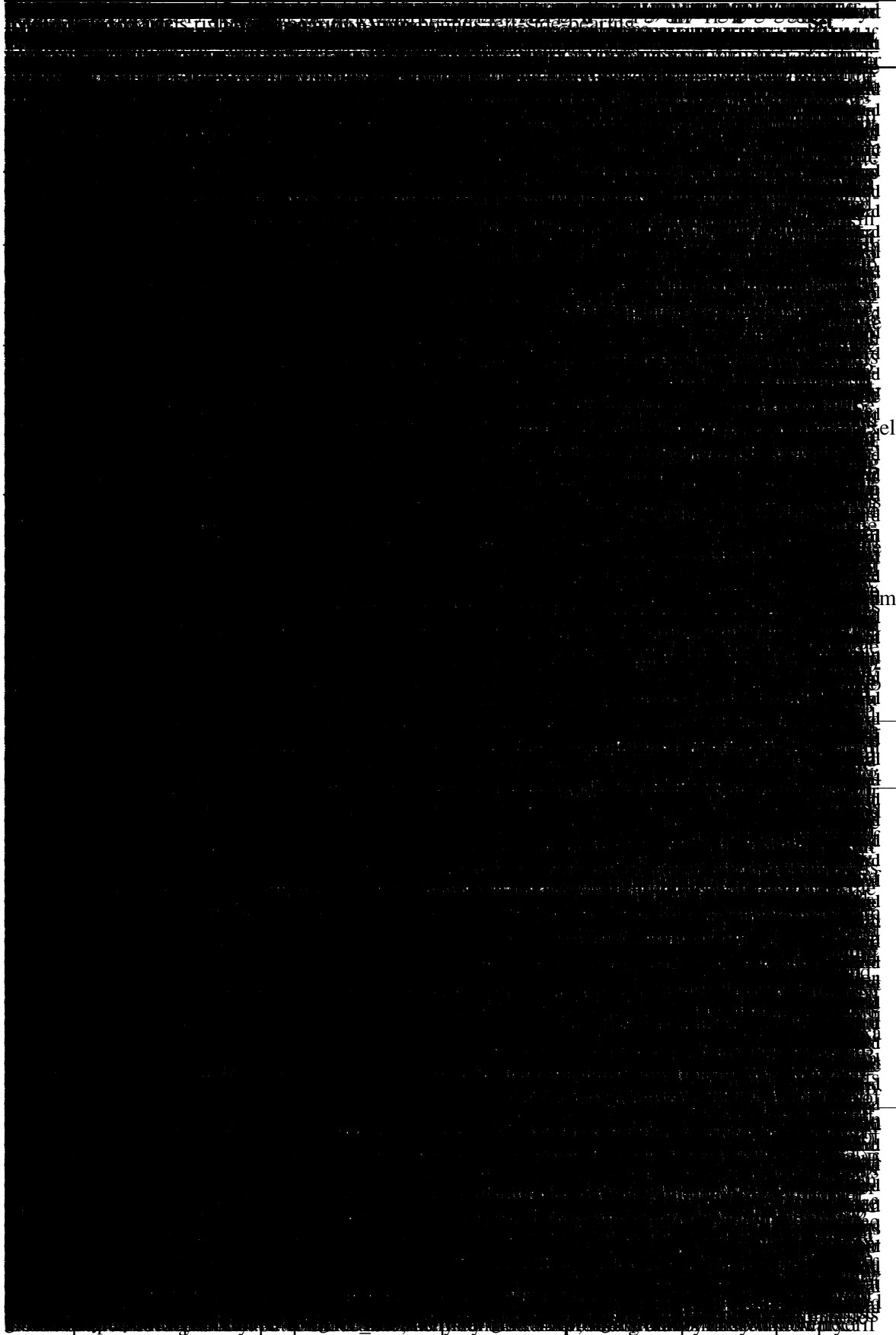
el
m-



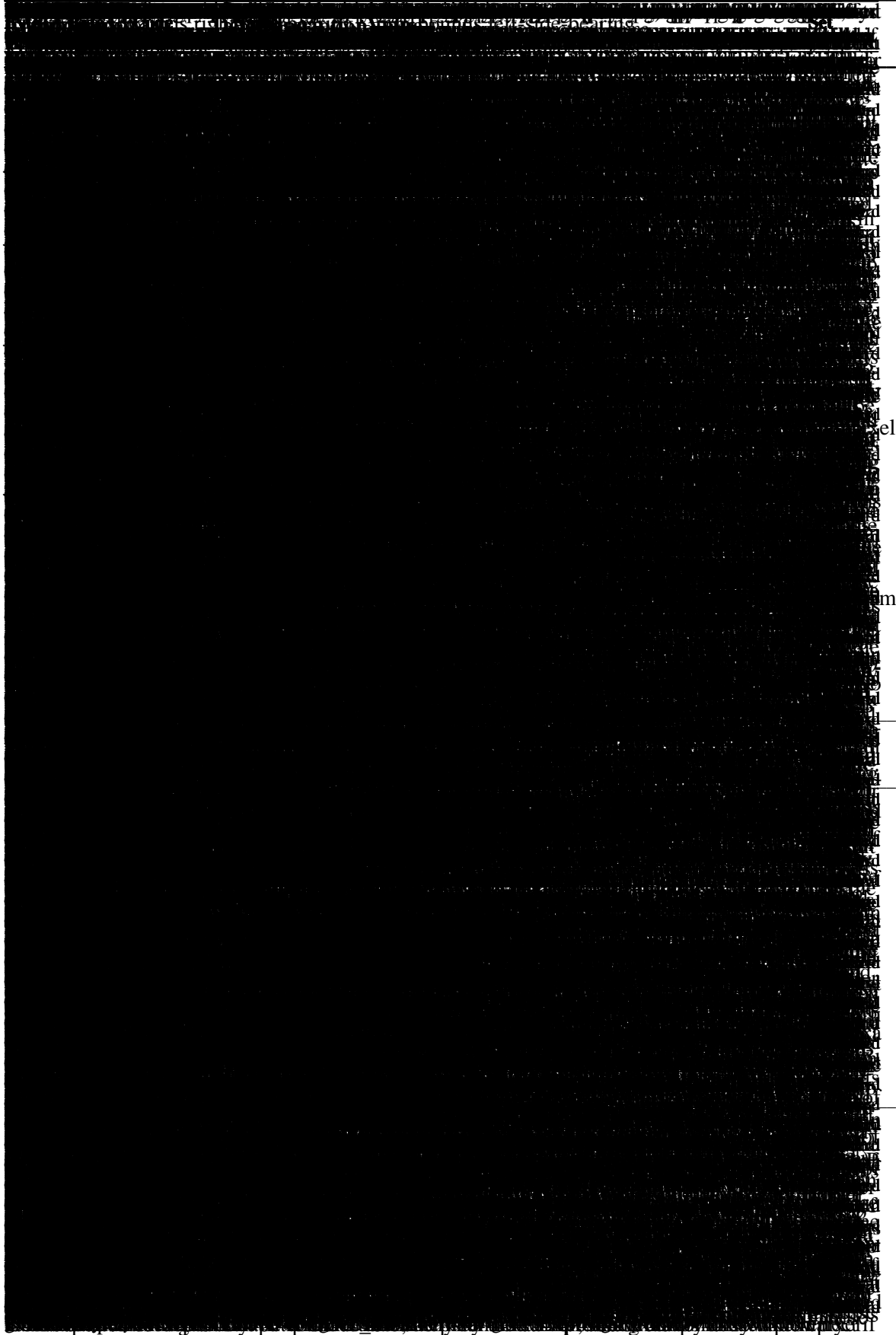
el
m-



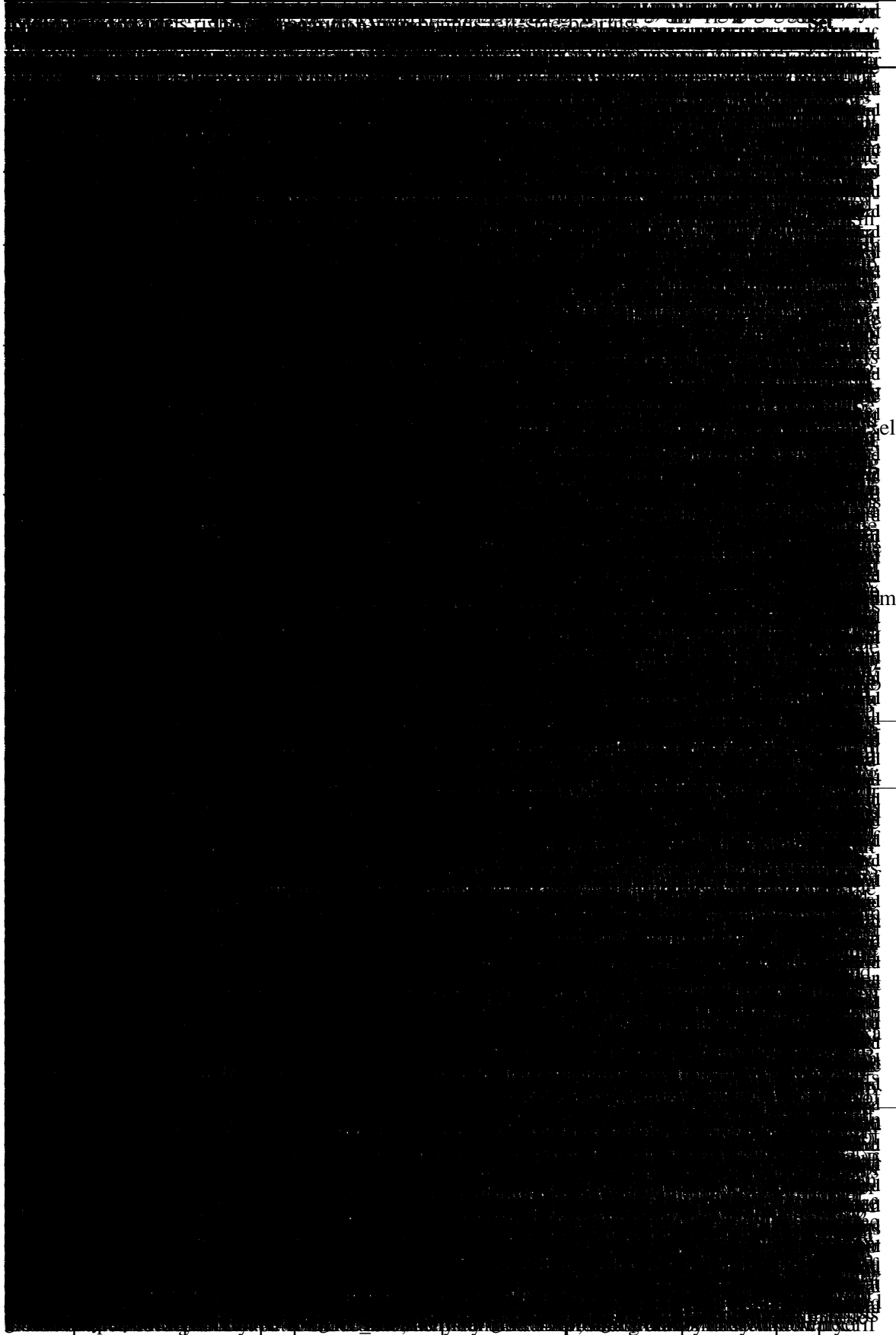
el
m-



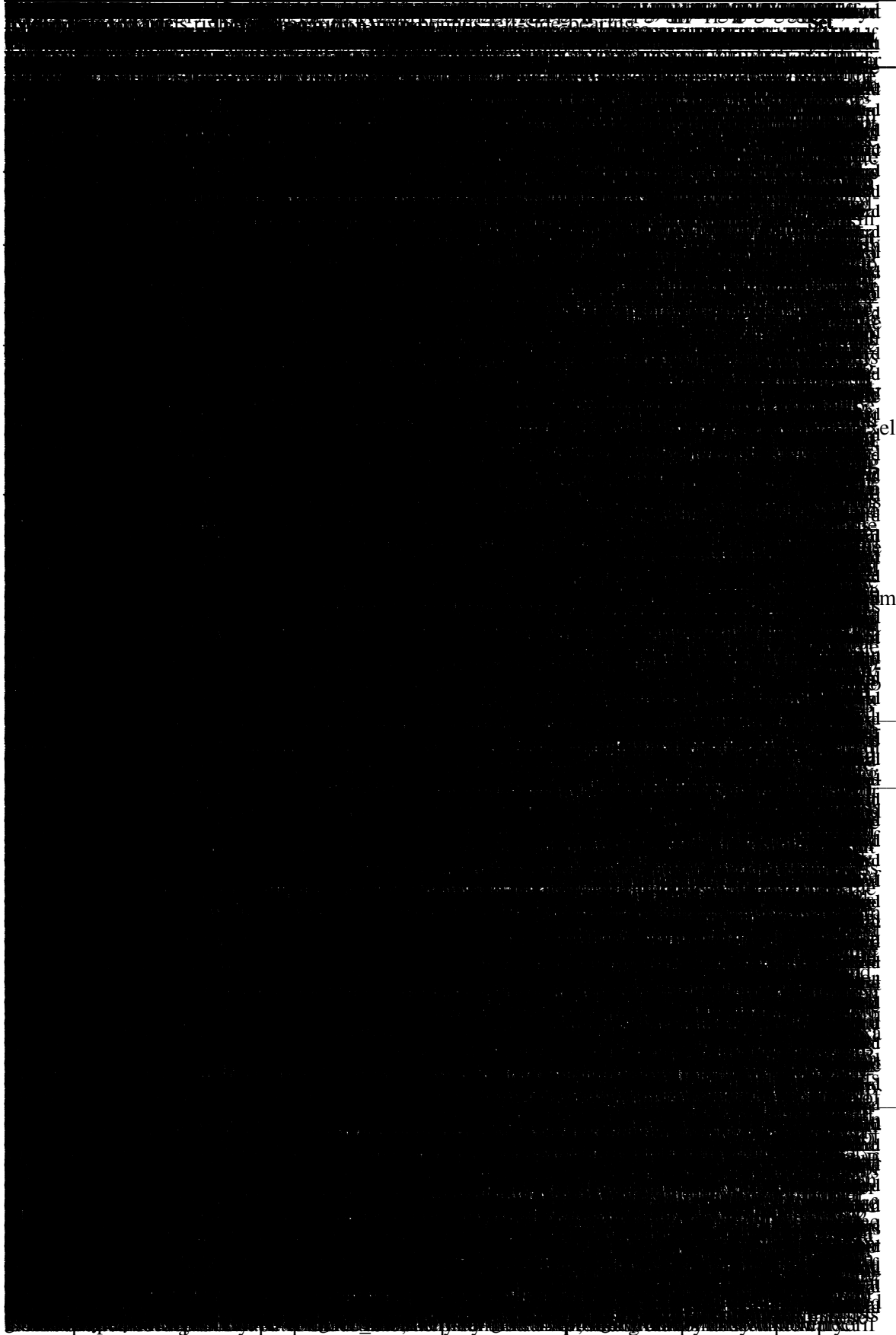
el
m-



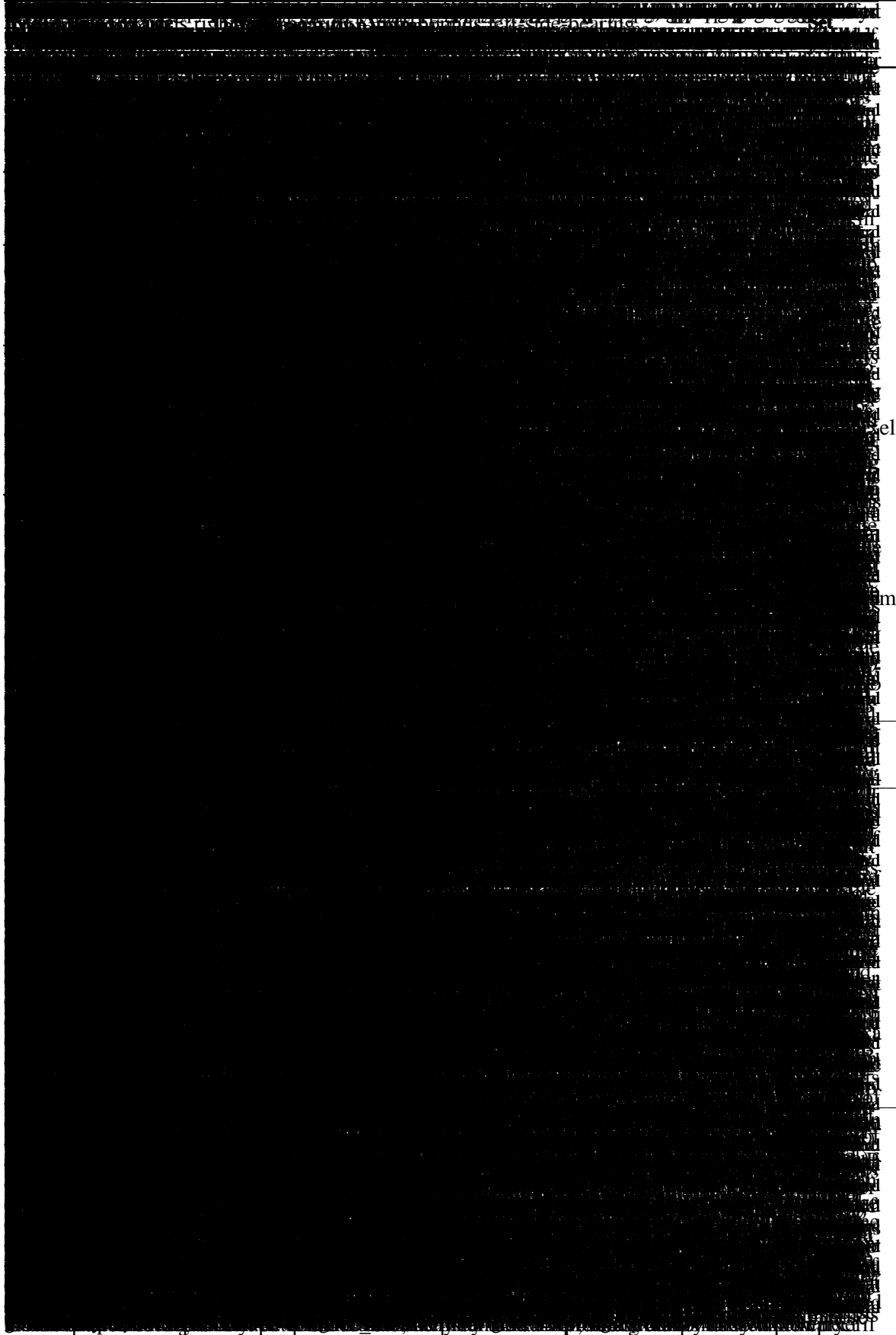
el
m-



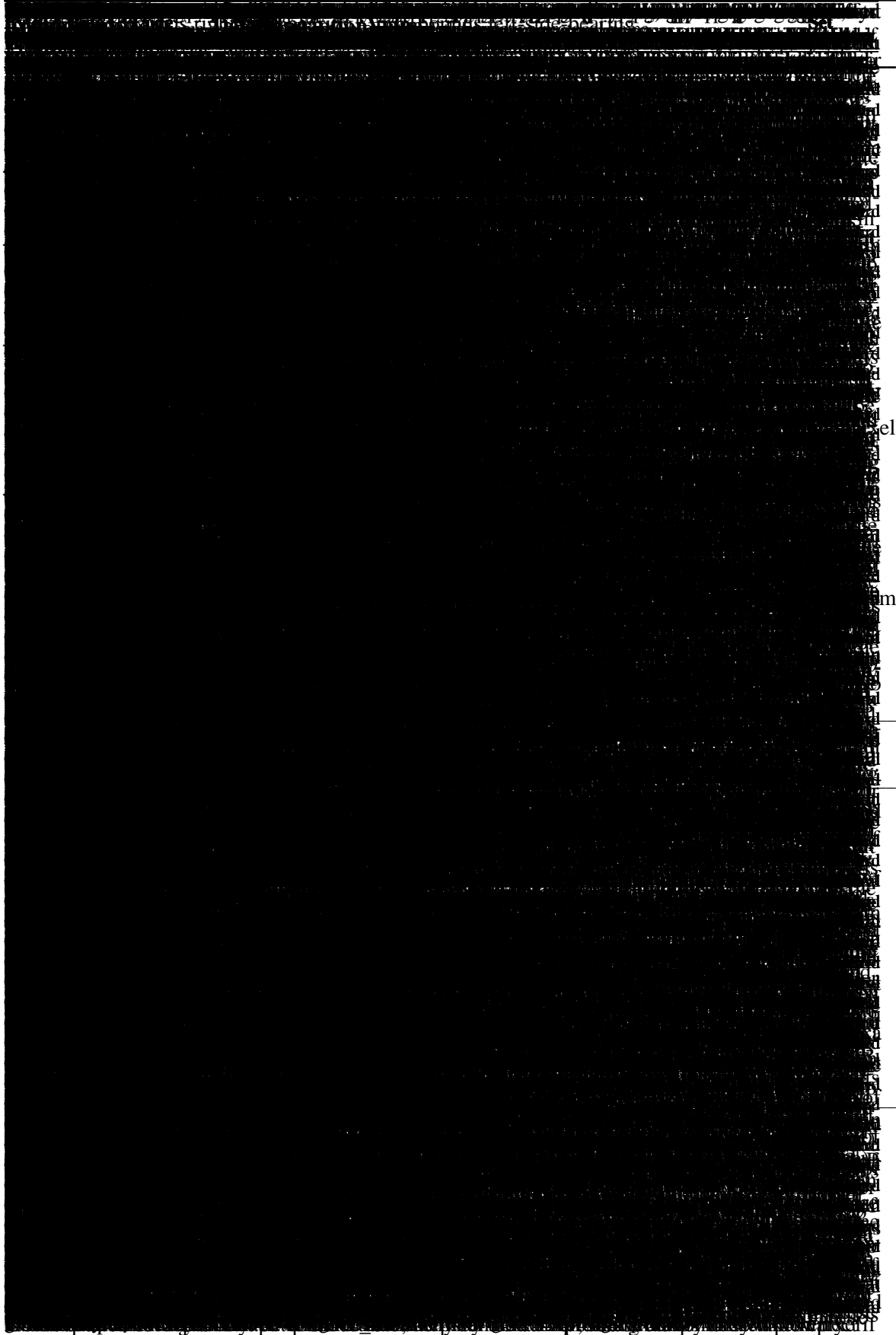
el
m-



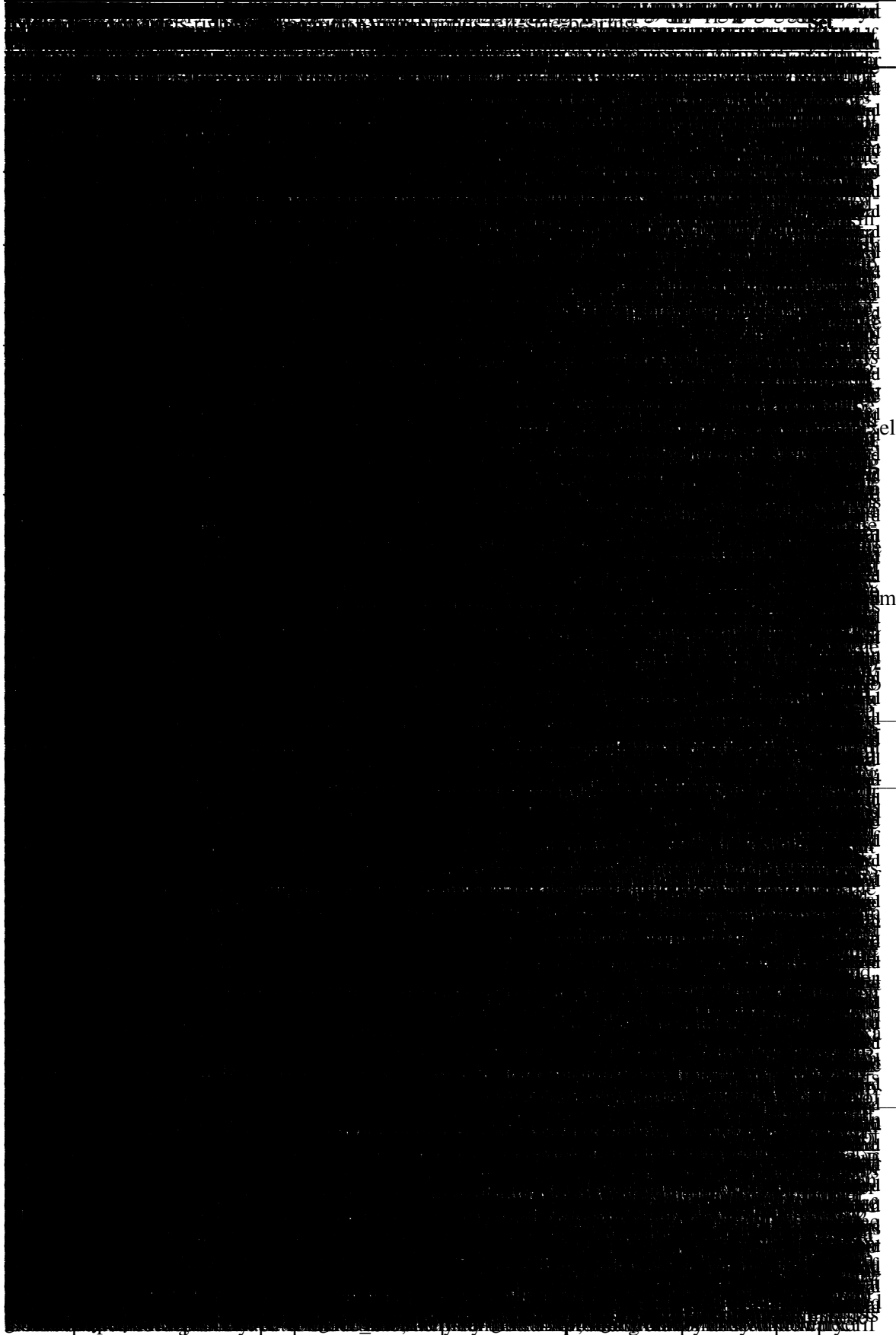
el
m-



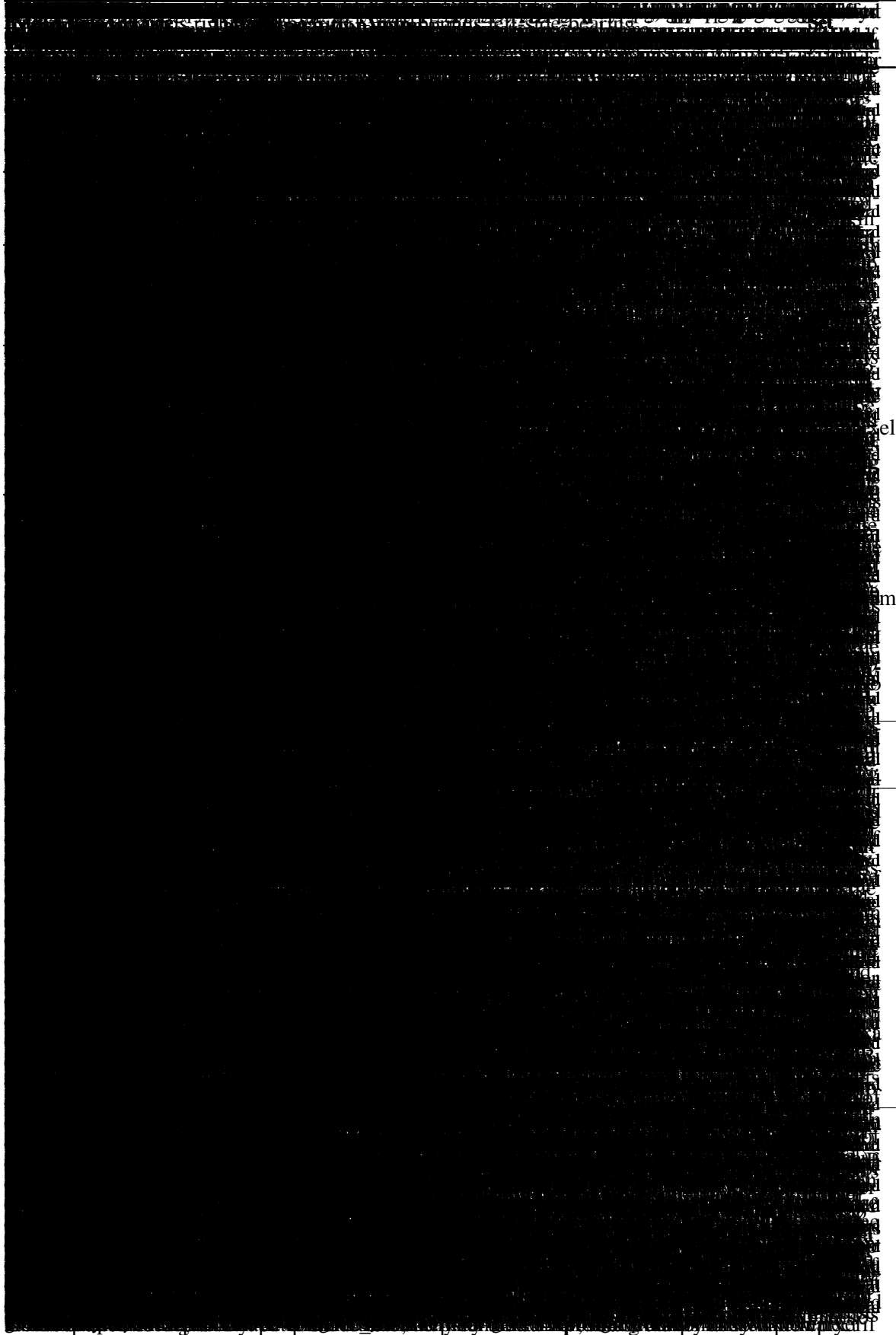
el
m-



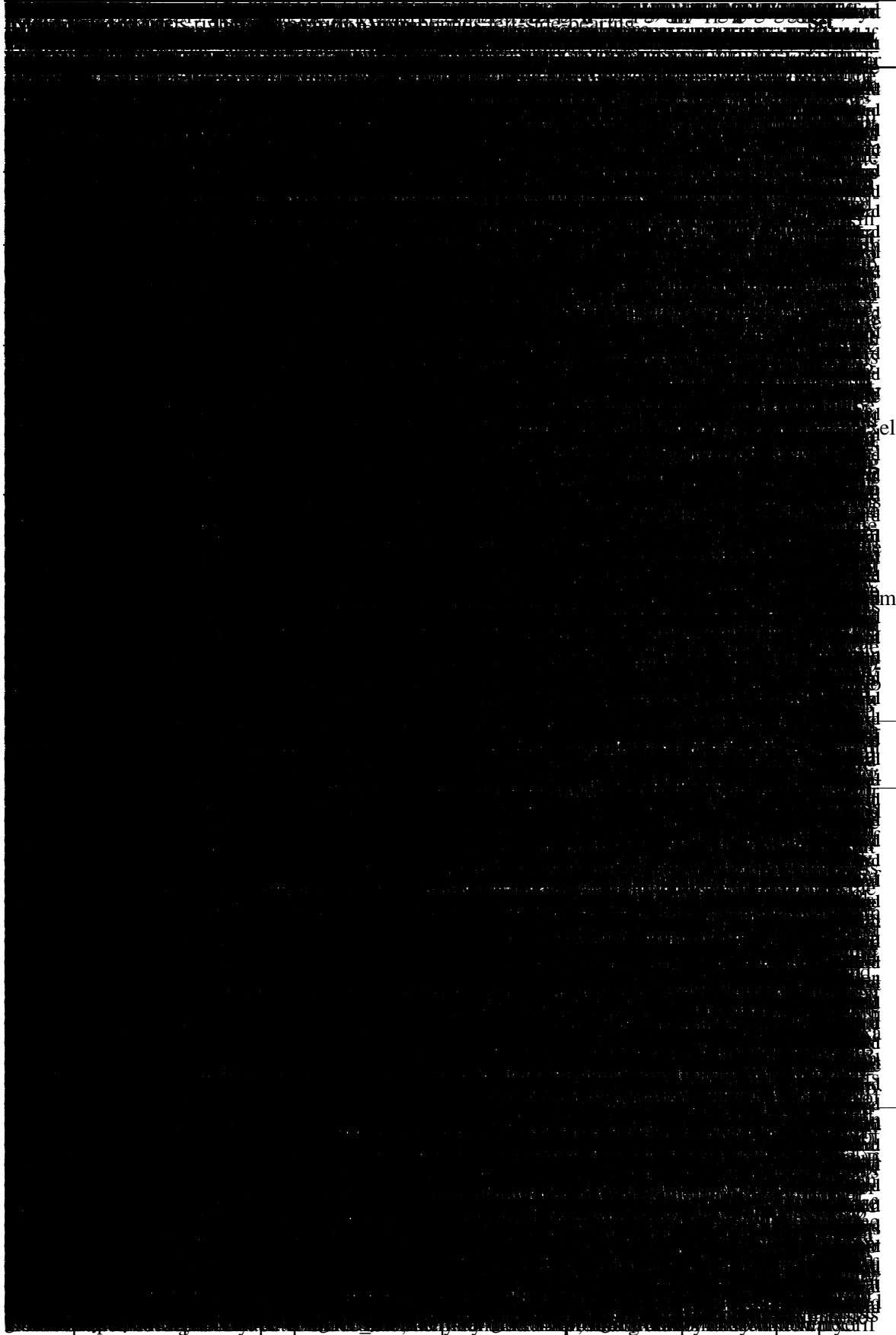
el
m-



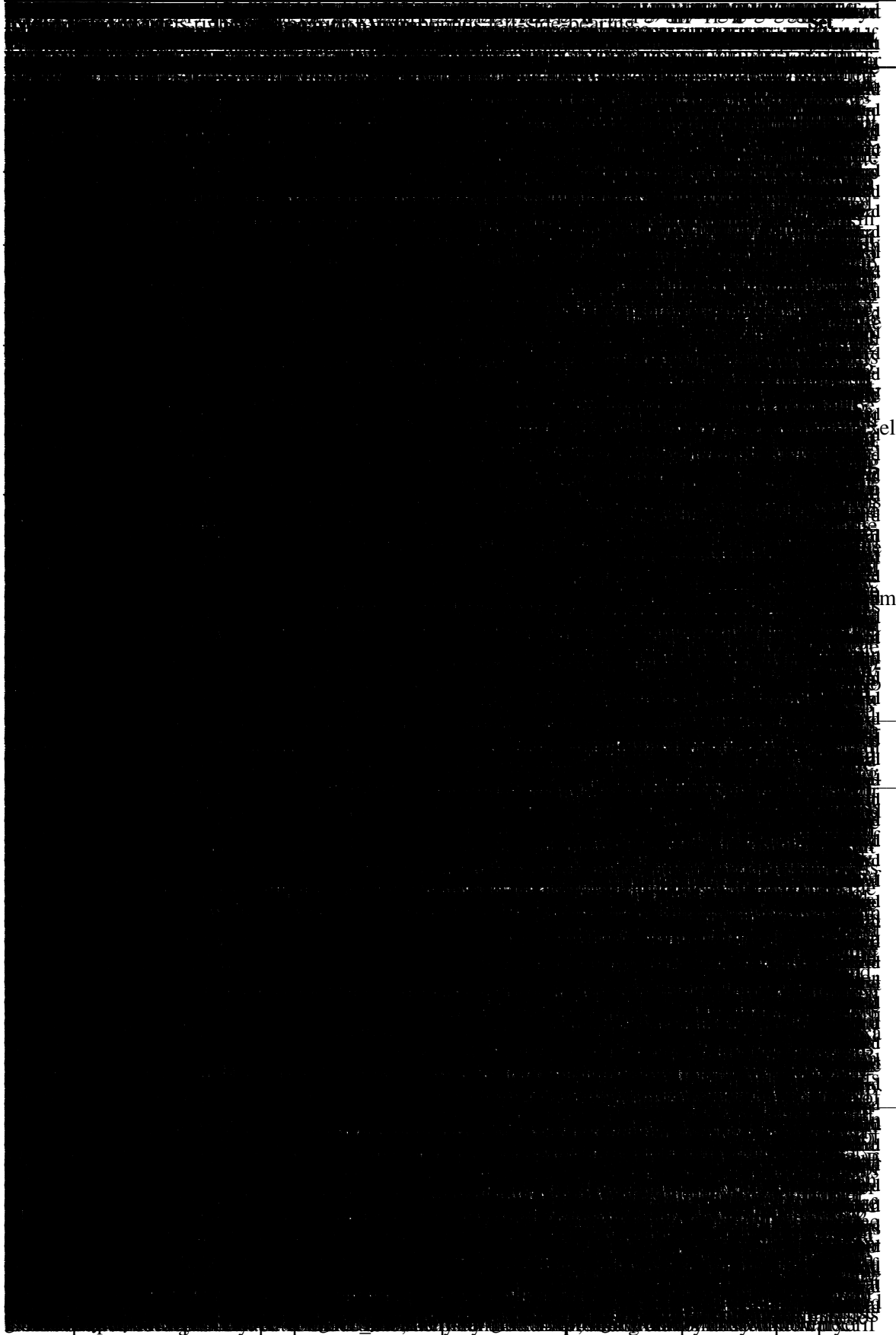
el
m-



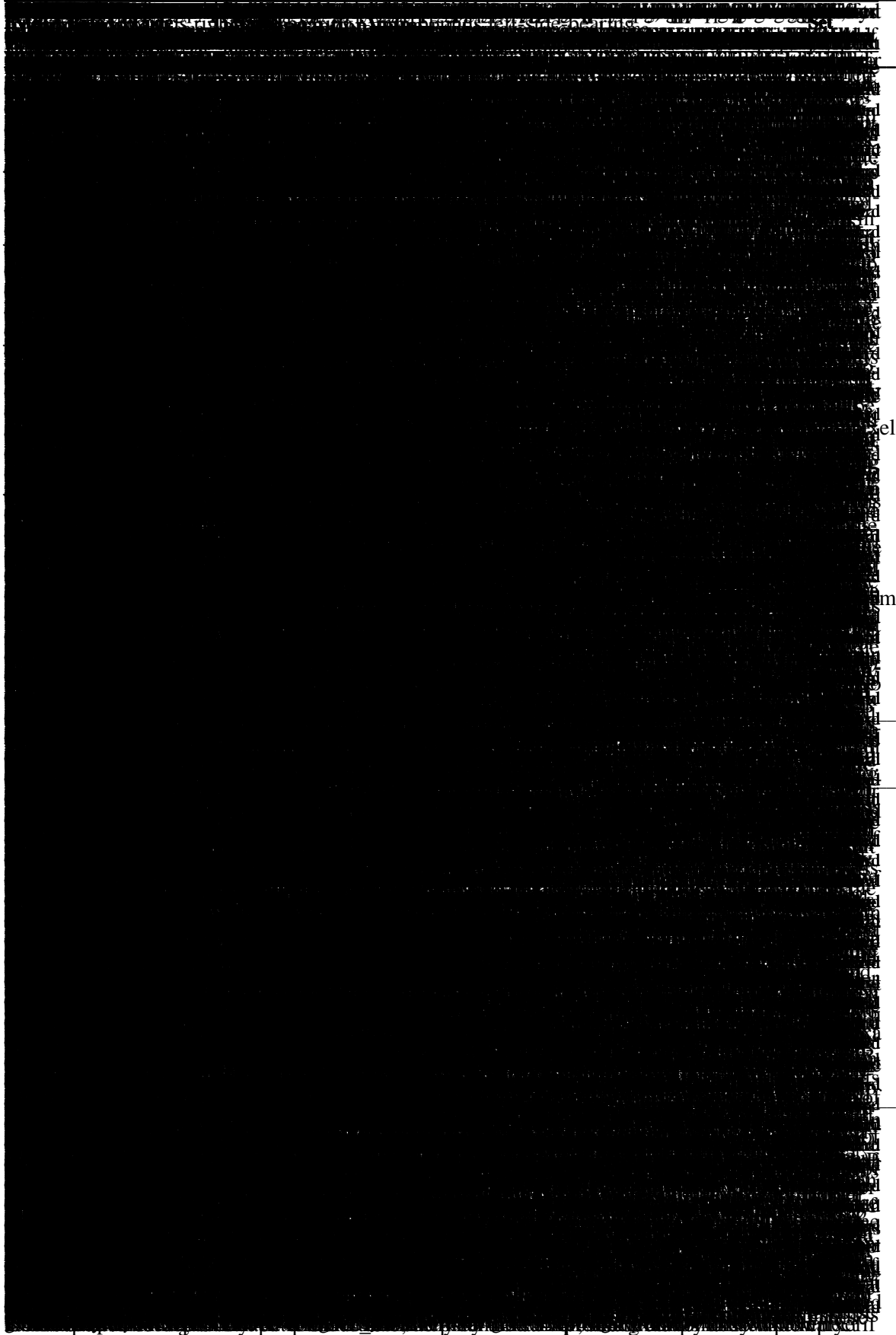
el
m-



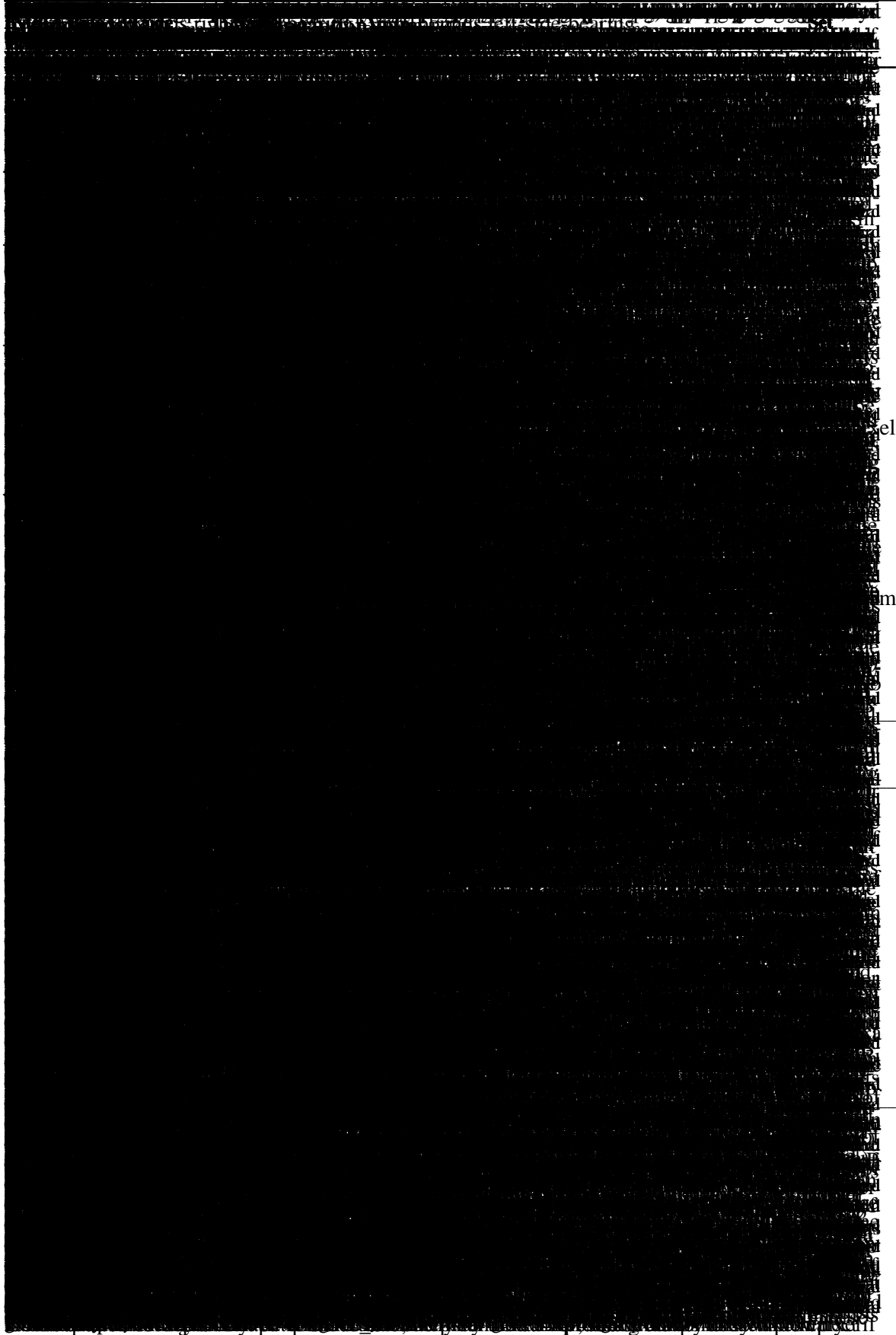
el
m-



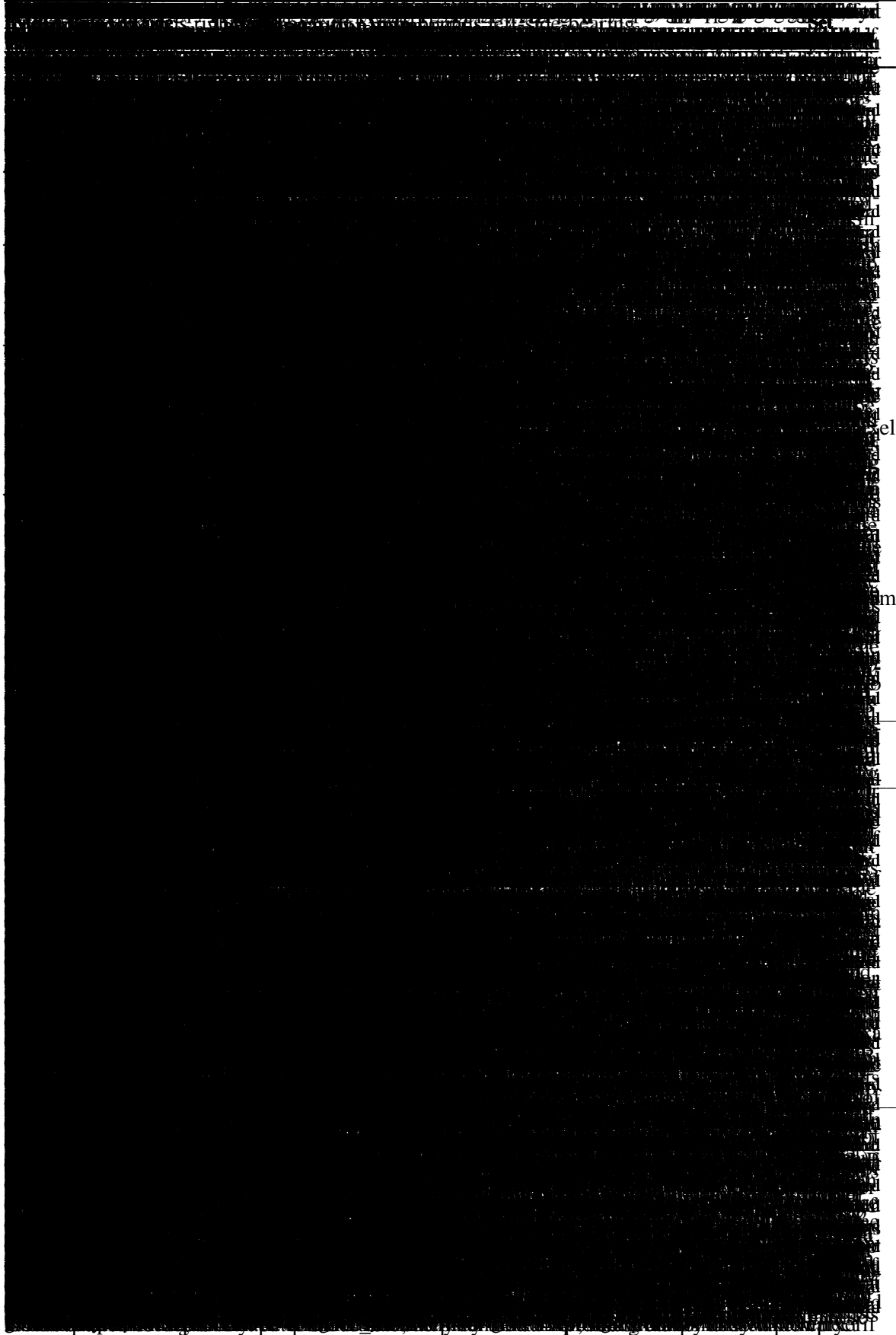
el
m-



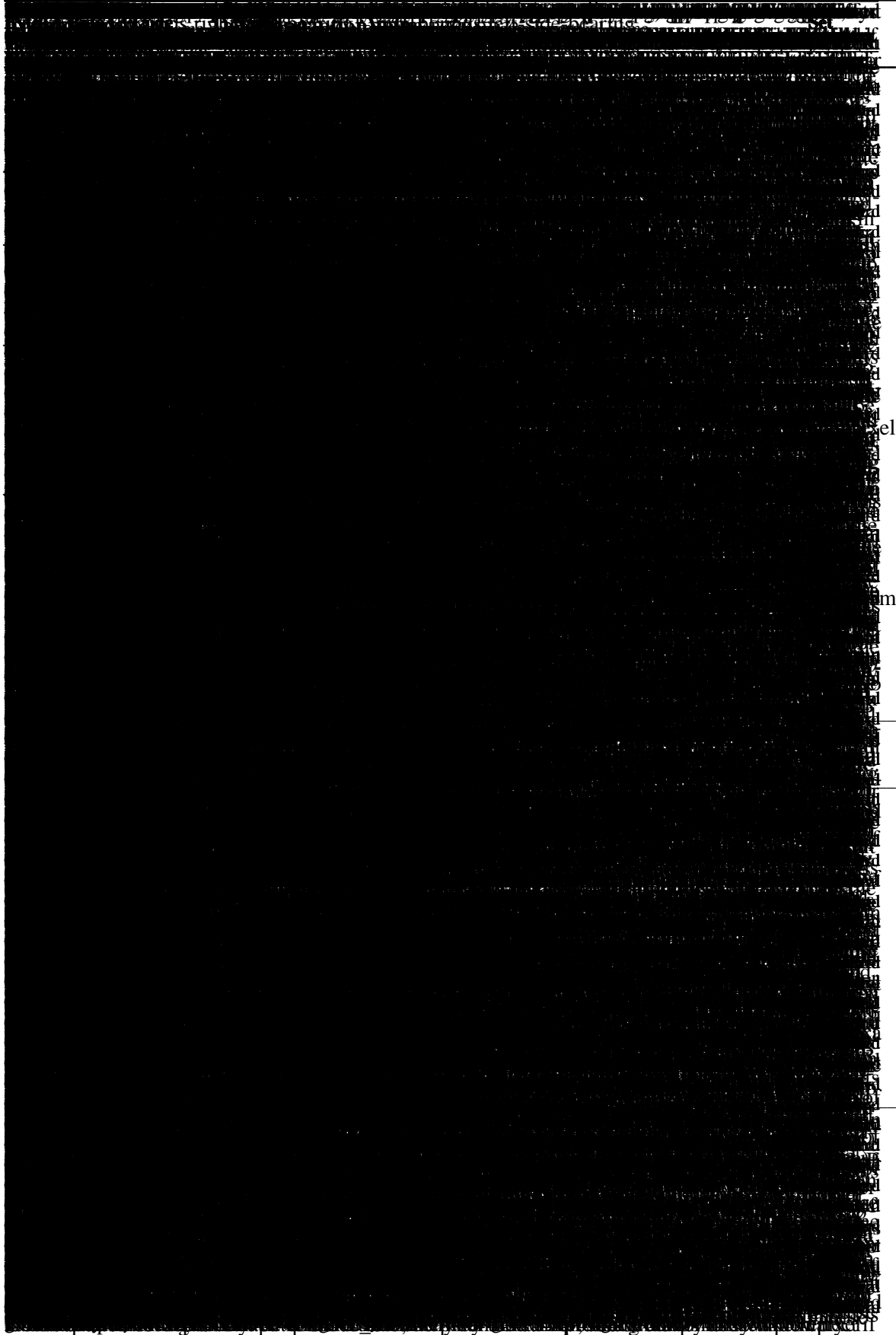
el
m-



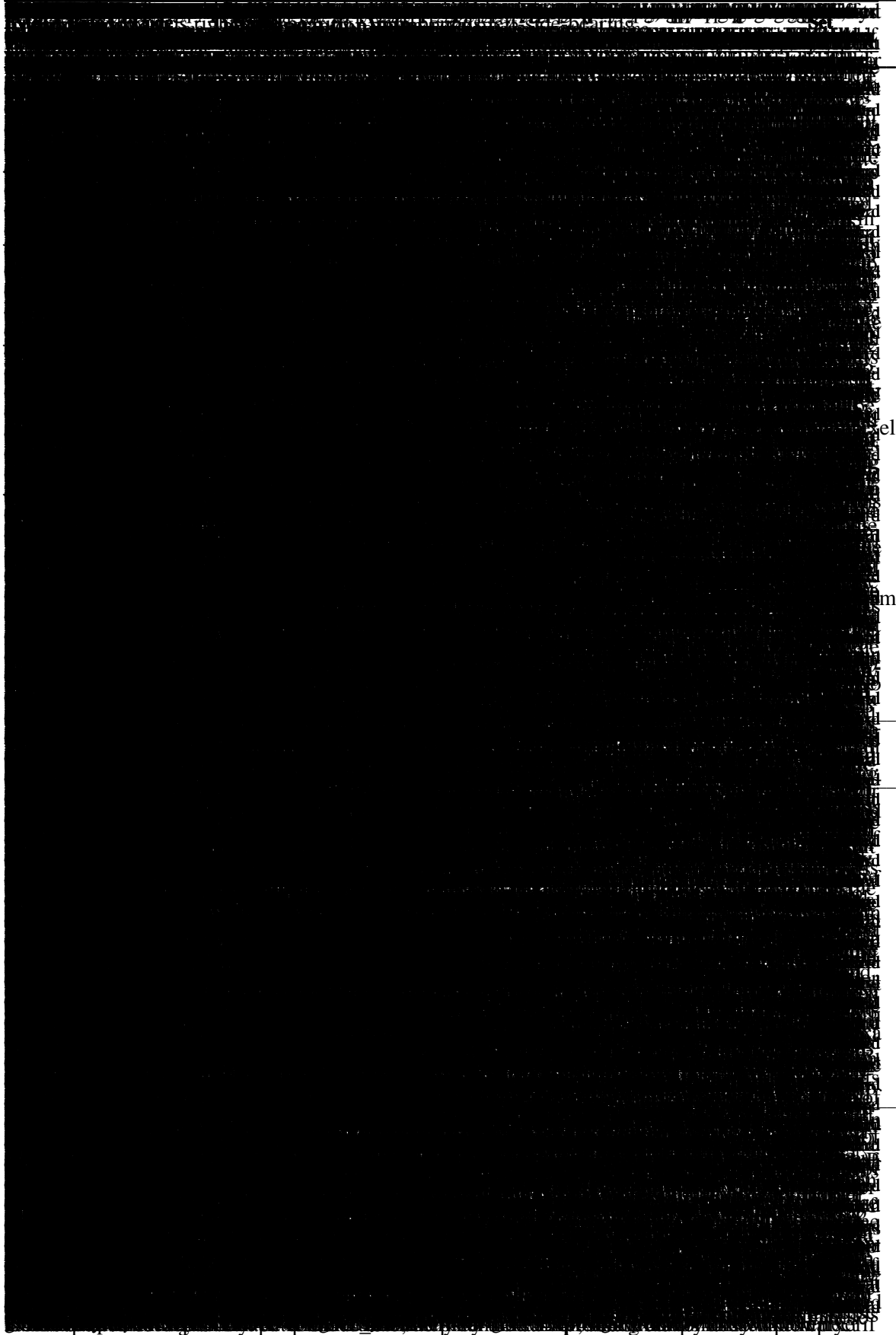
el
m-



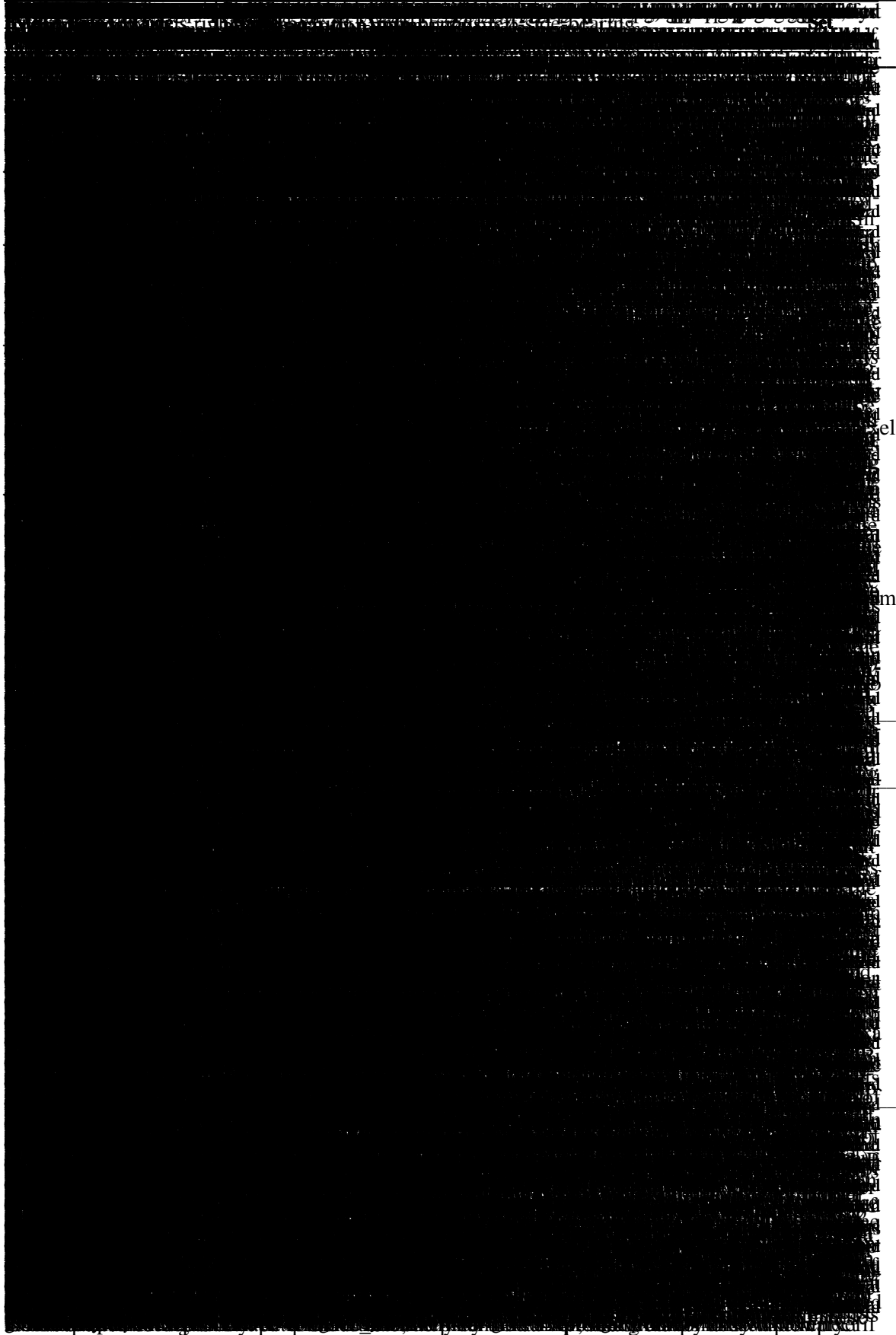
el
m-



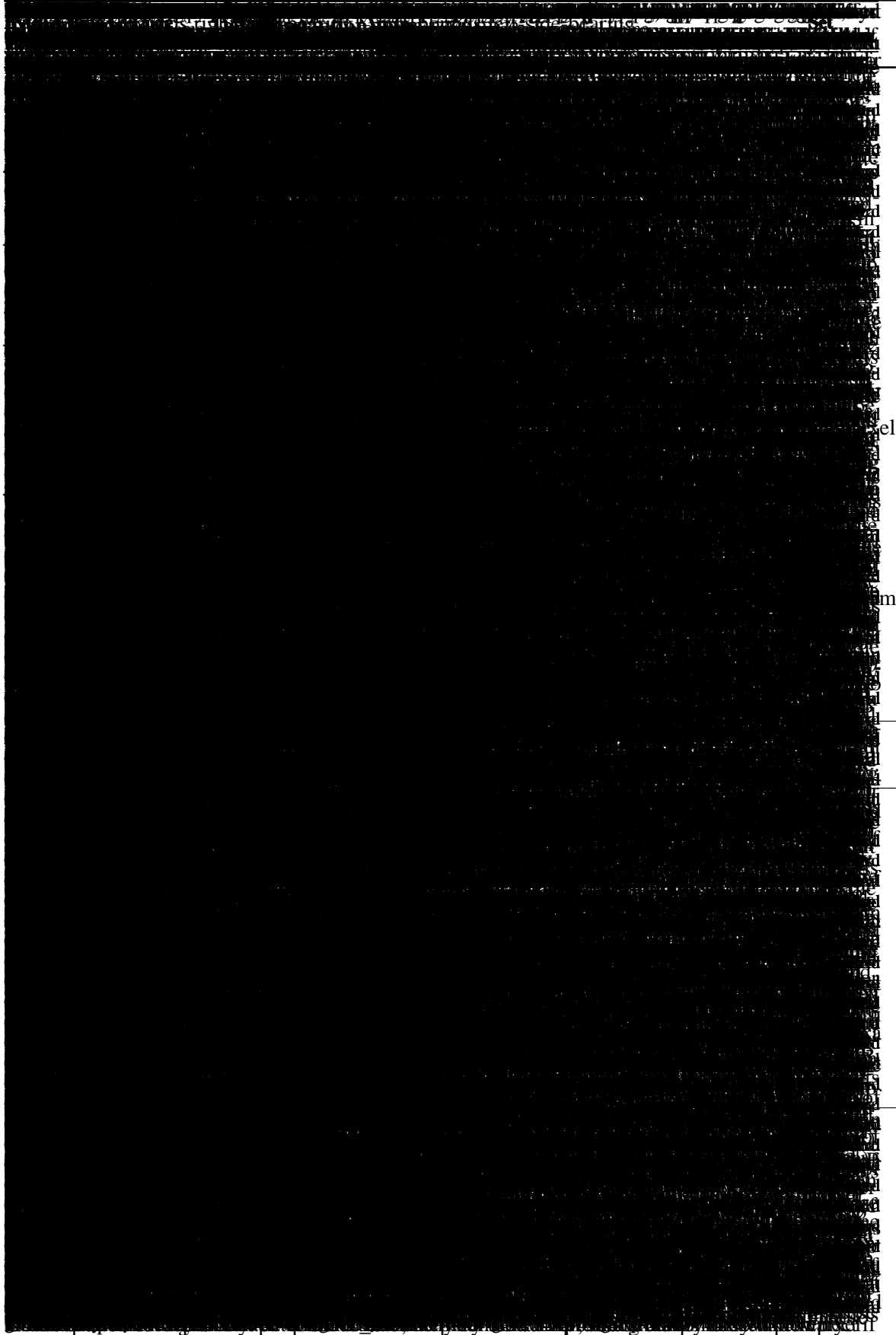
el
m-



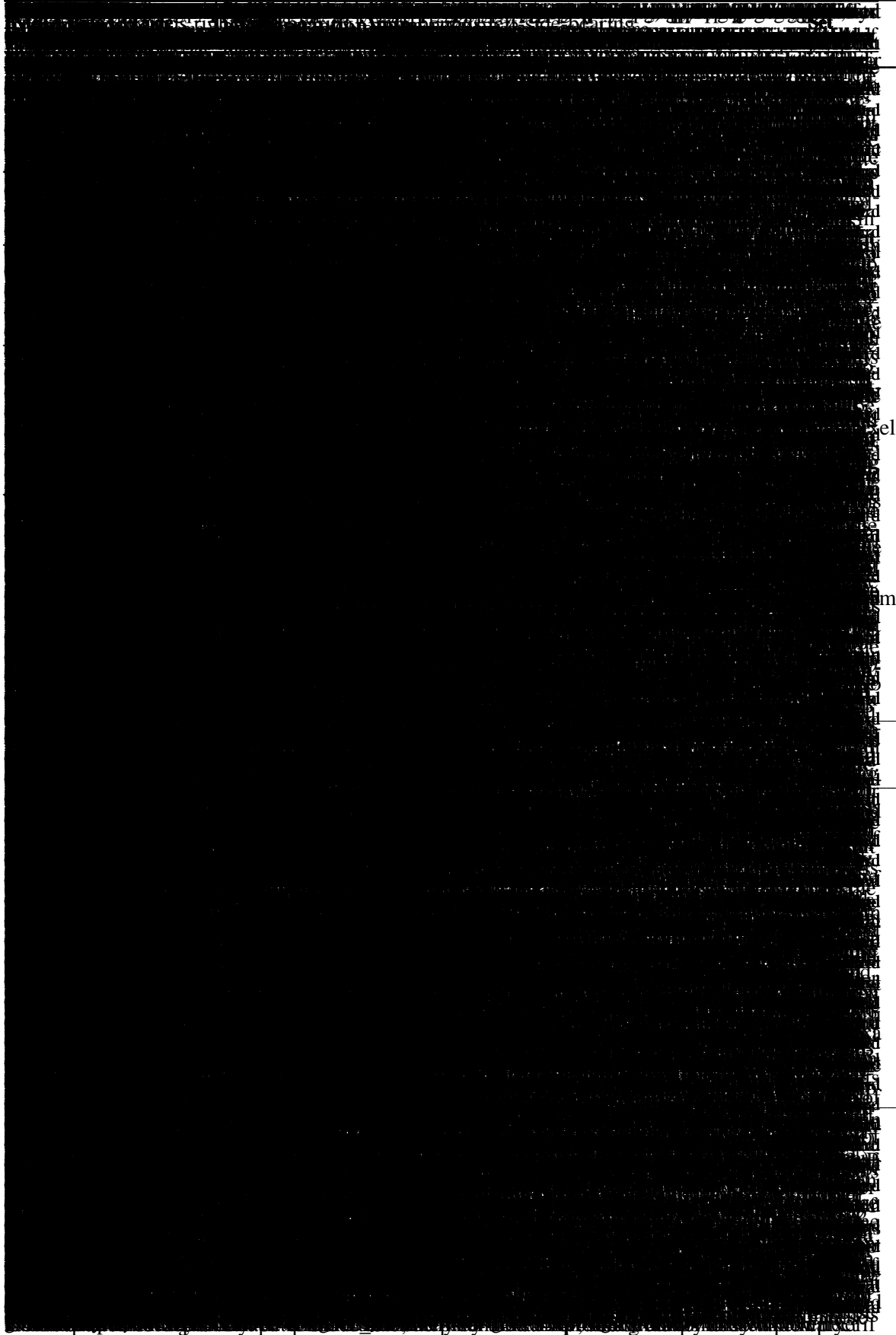
el
m-



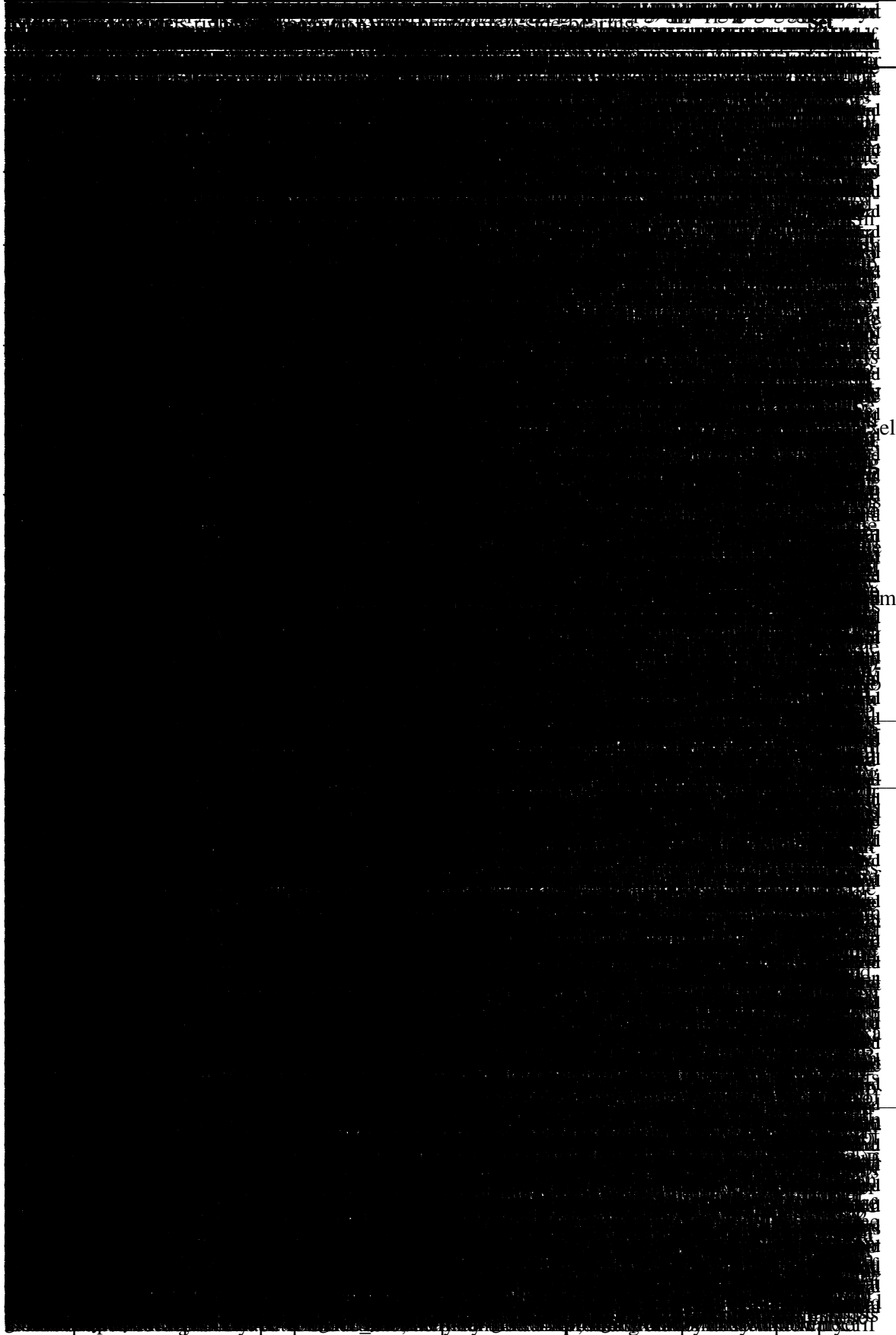
el
m-



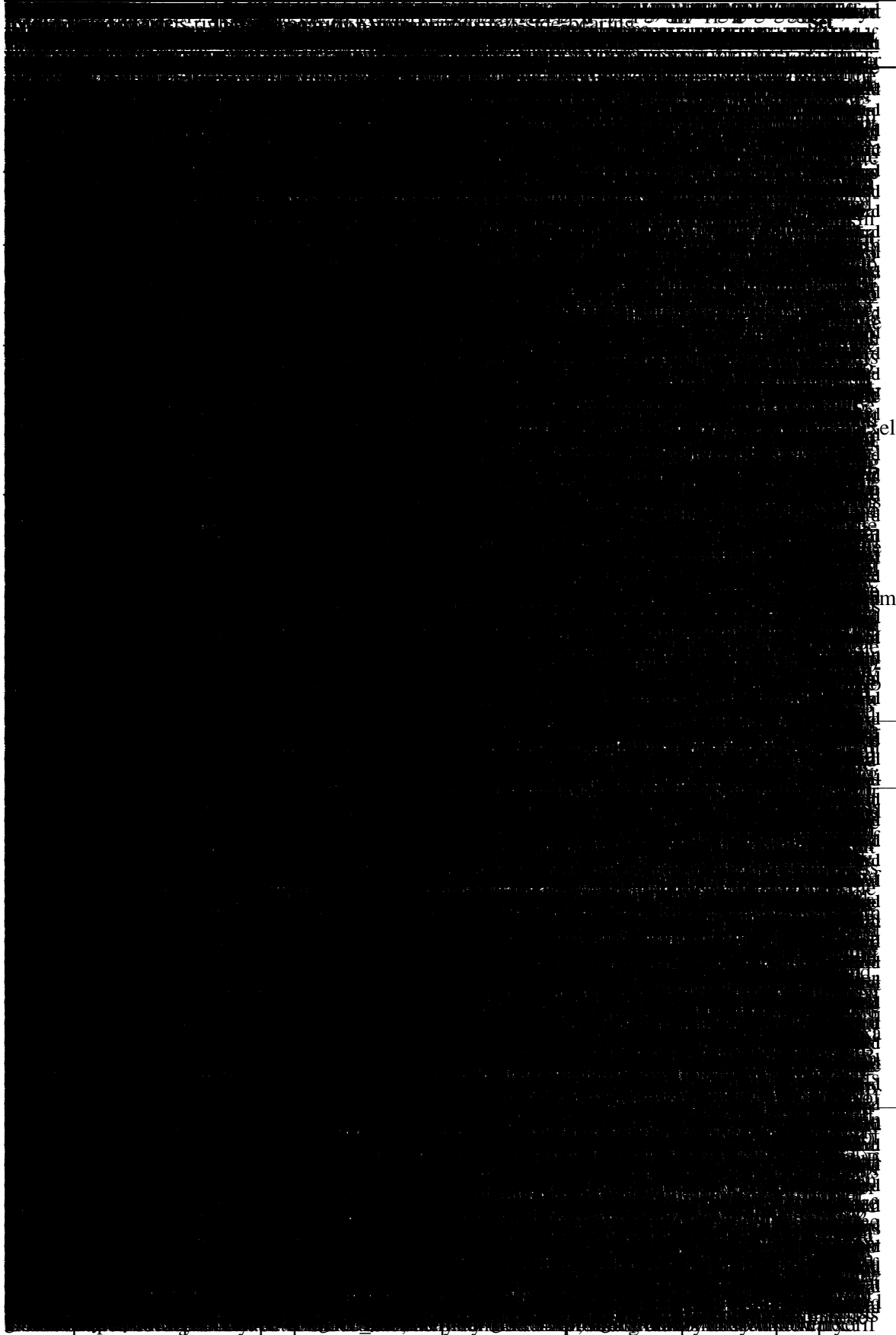
el
m-



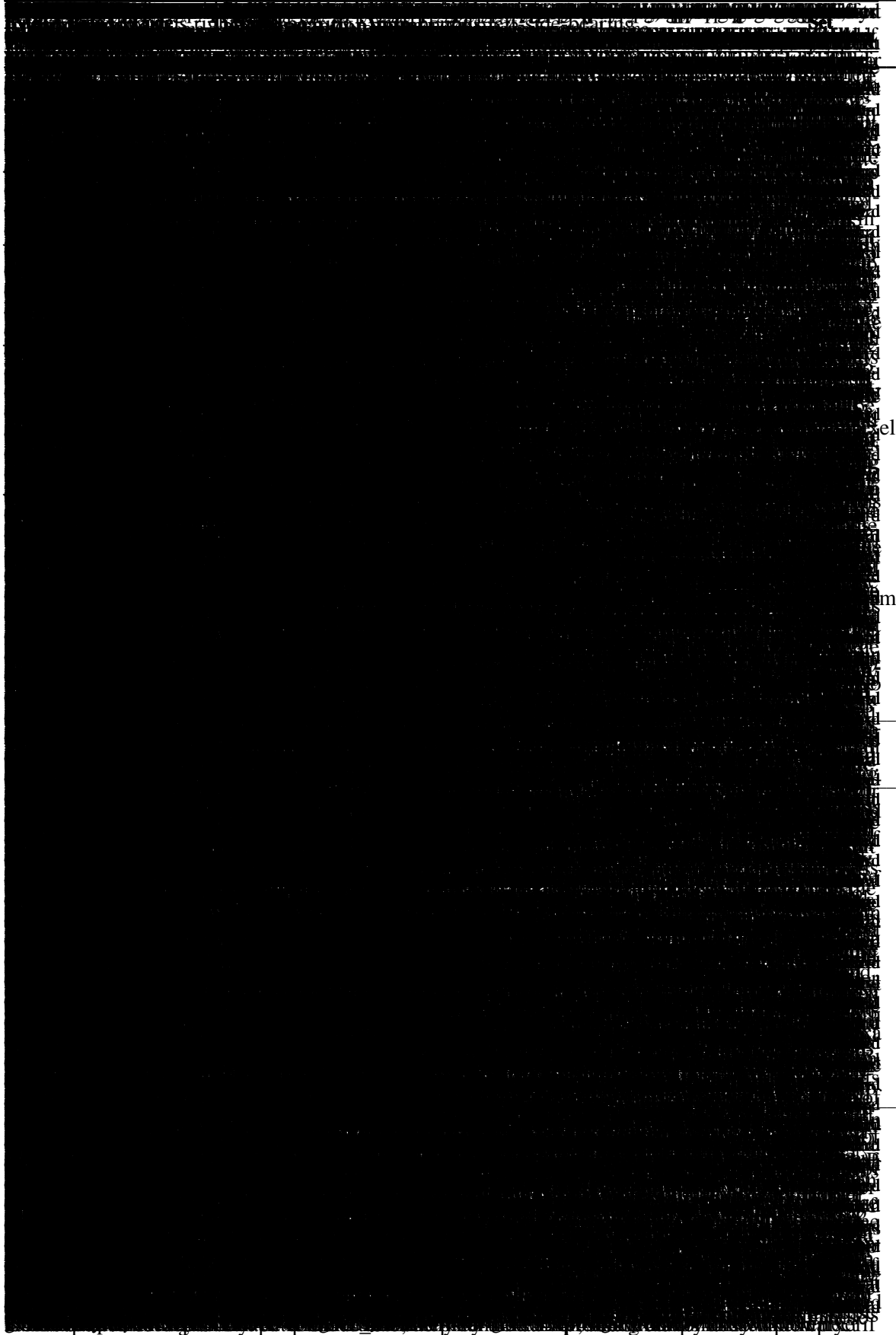
el
m-



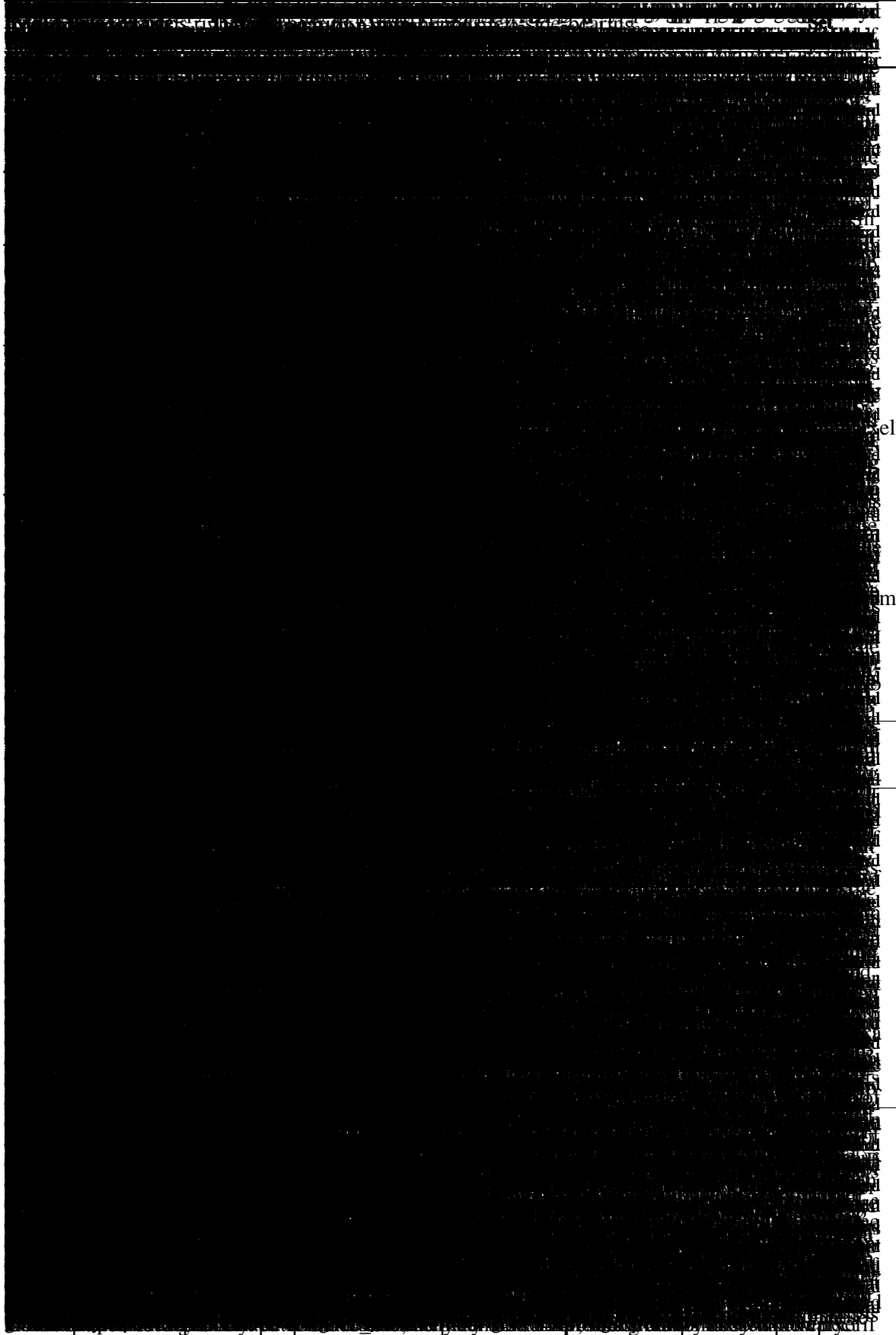
el
m-



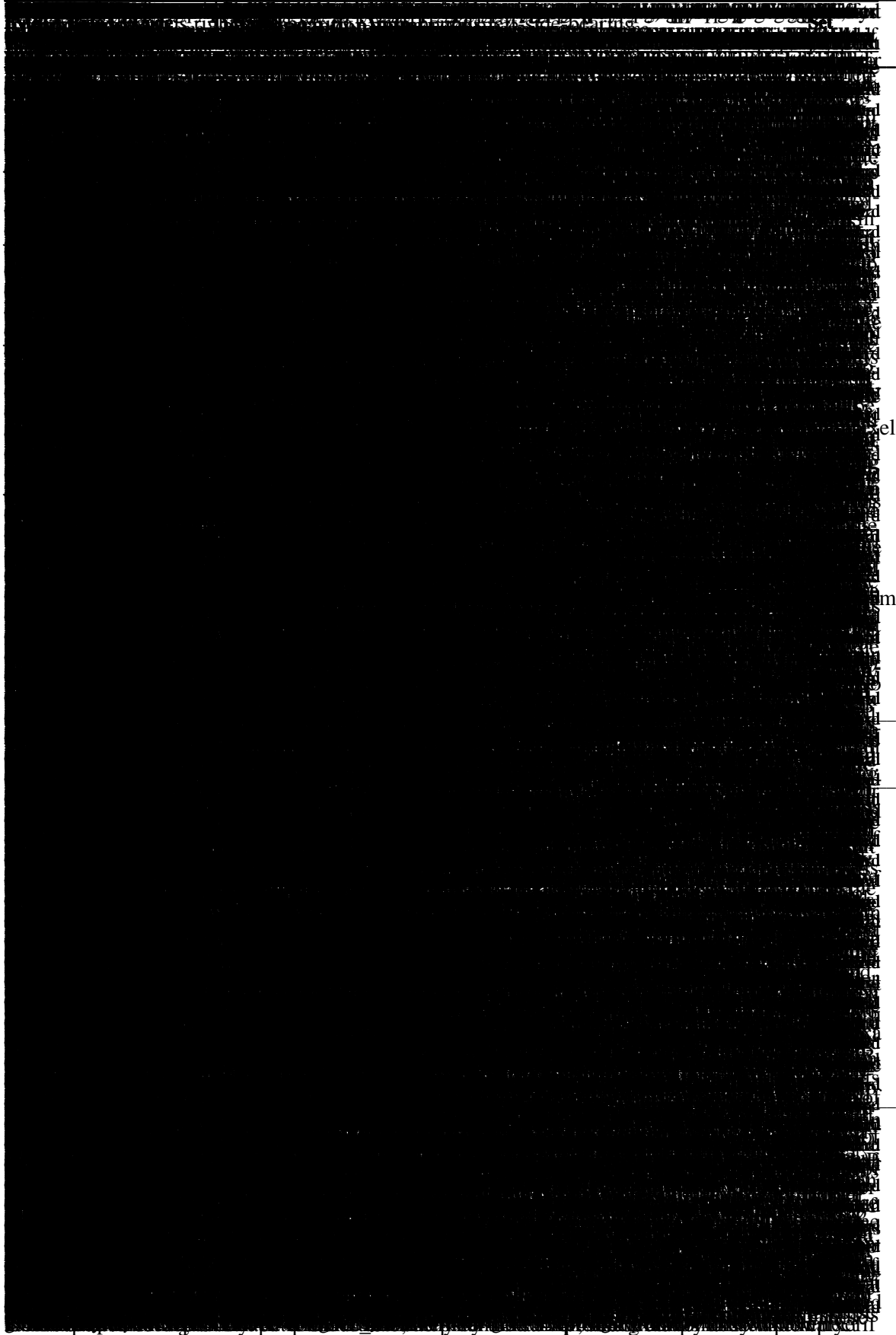
el
m-



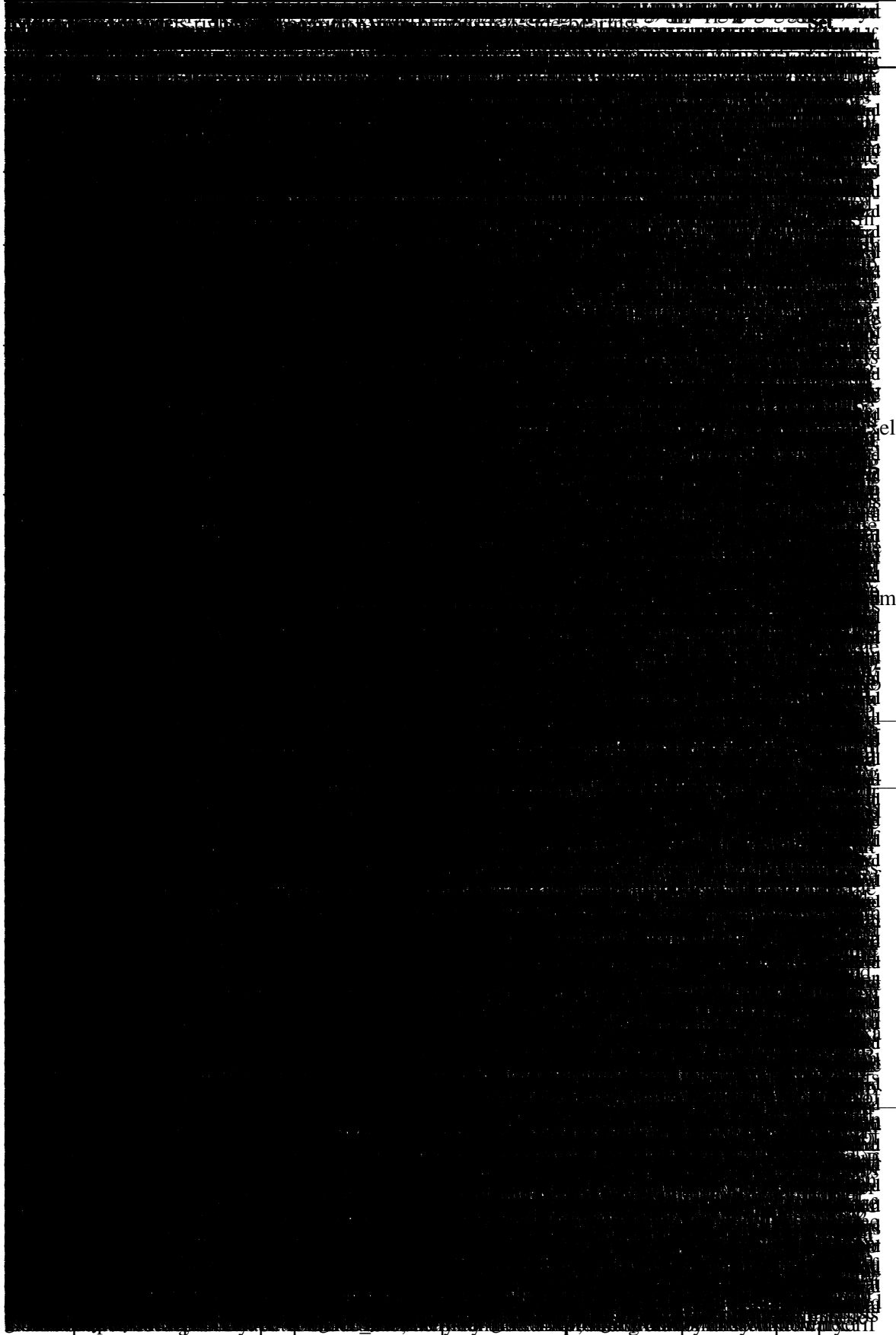
el
m-



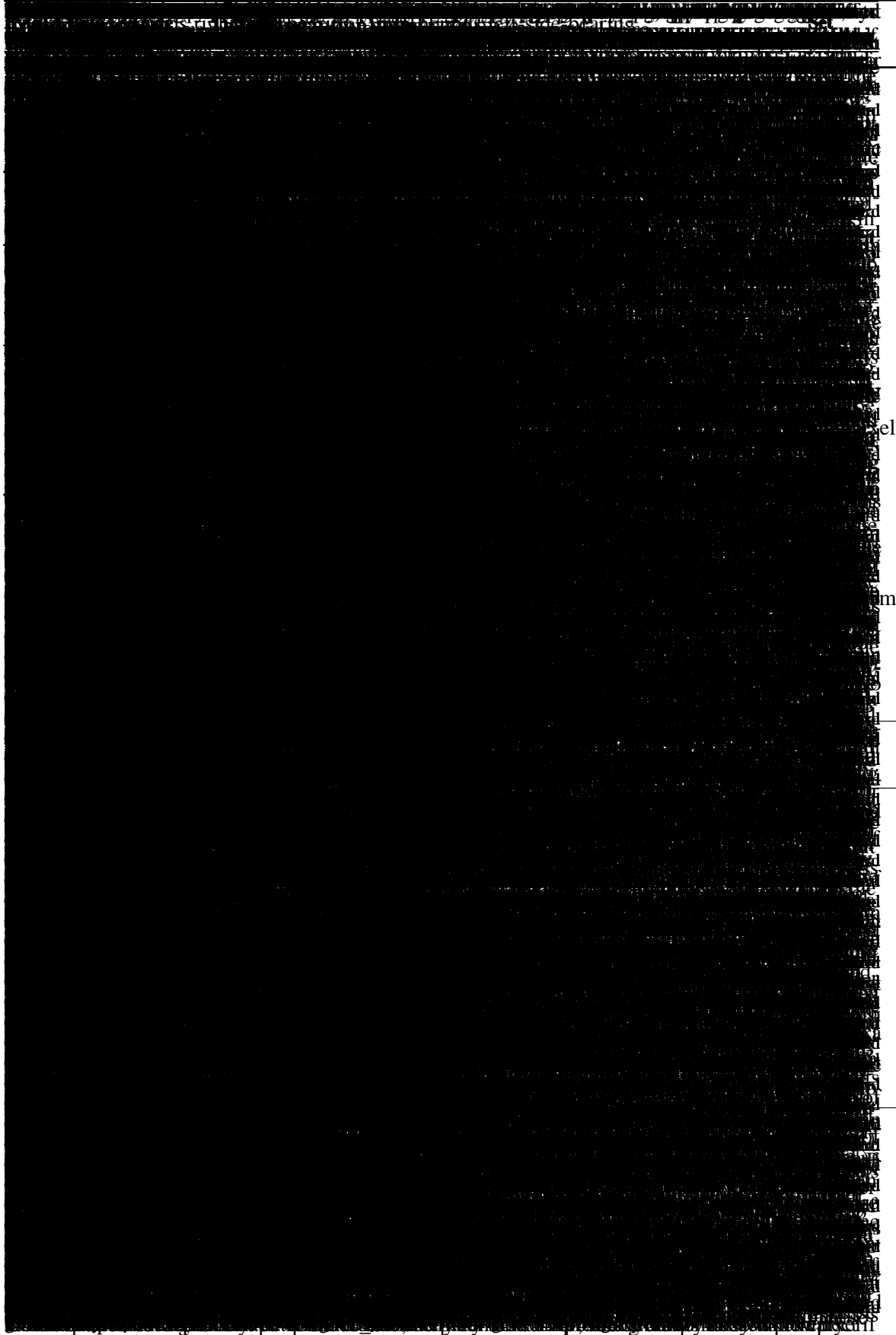
el
m-



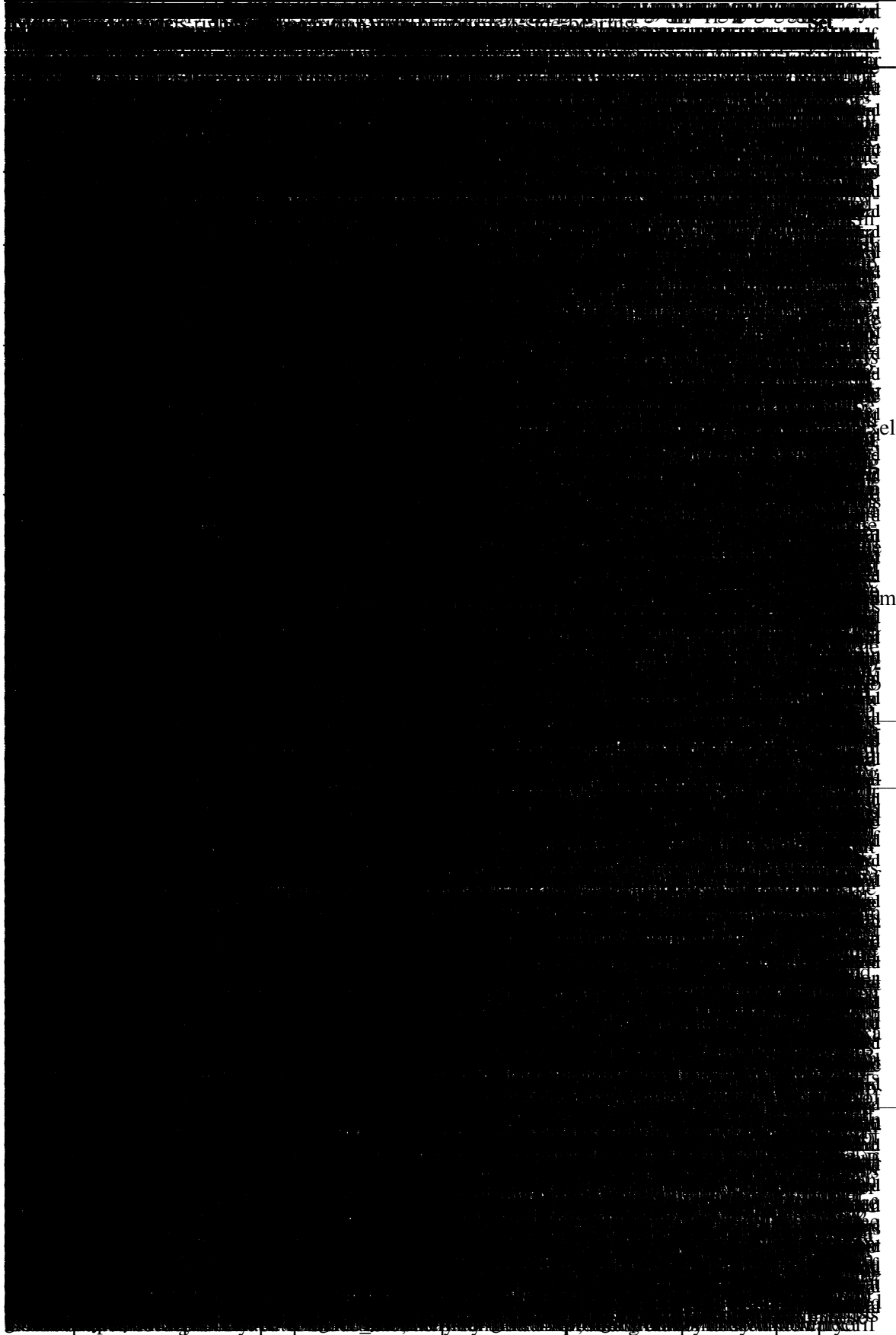
el
m-



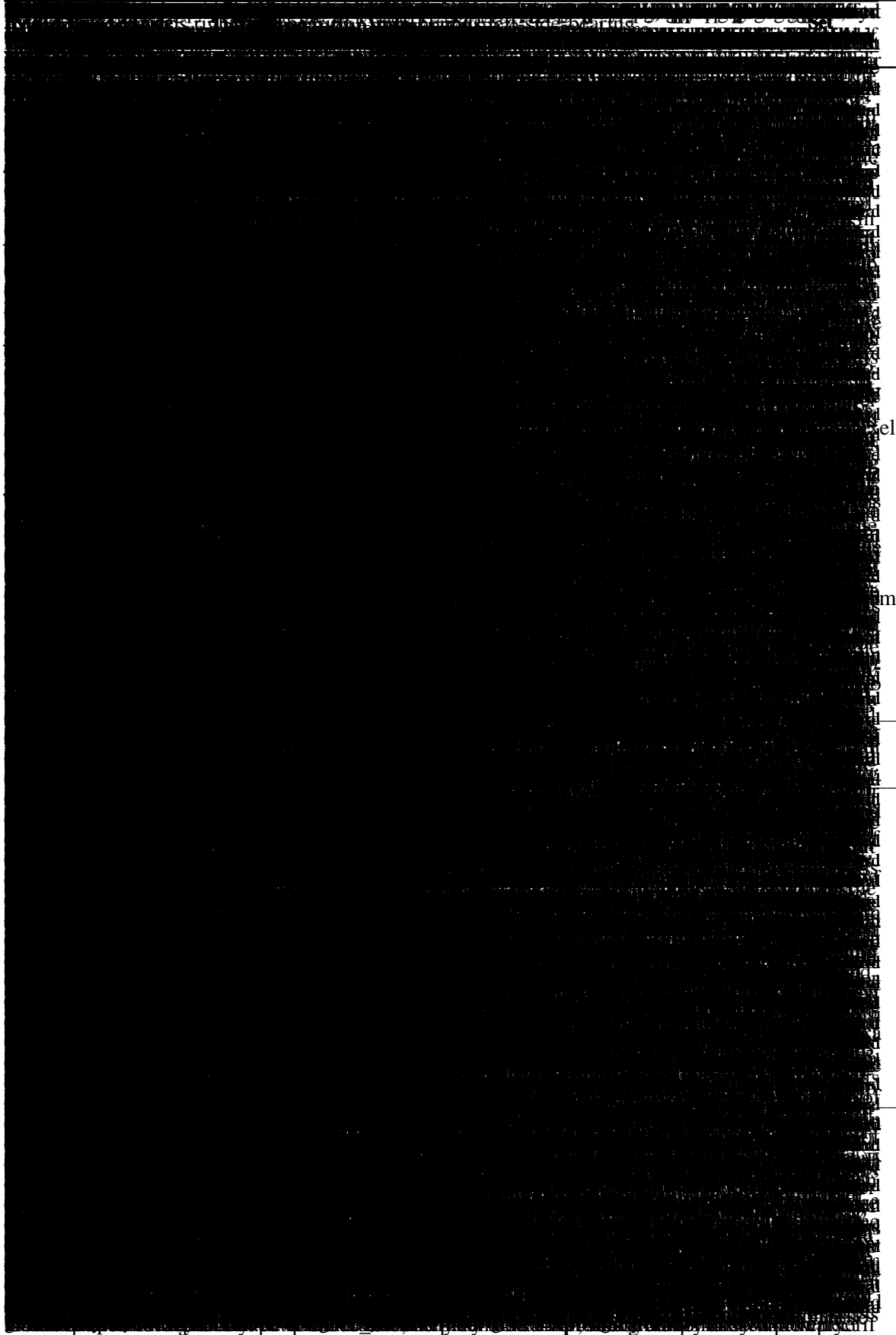
el
m-



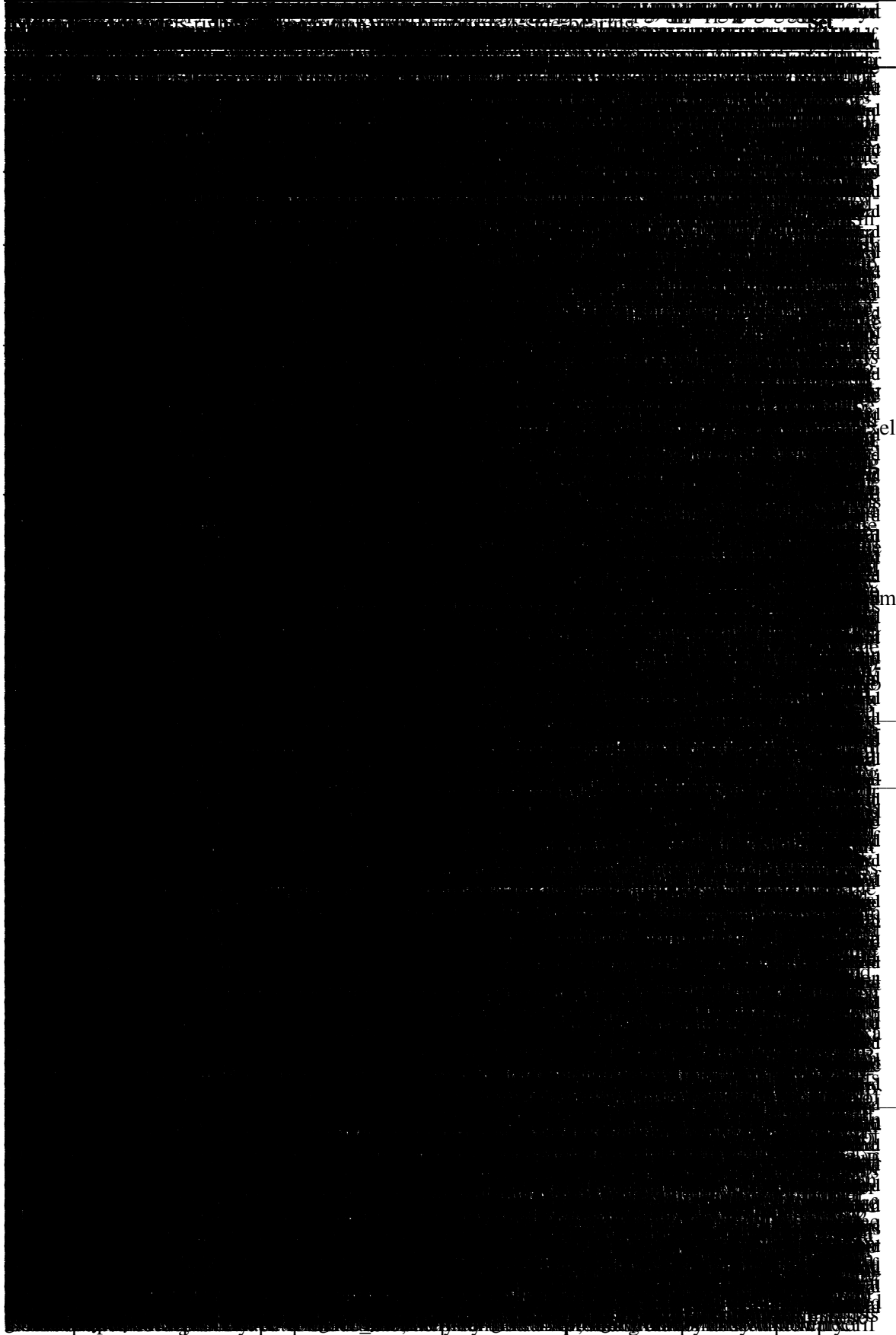
el
m-



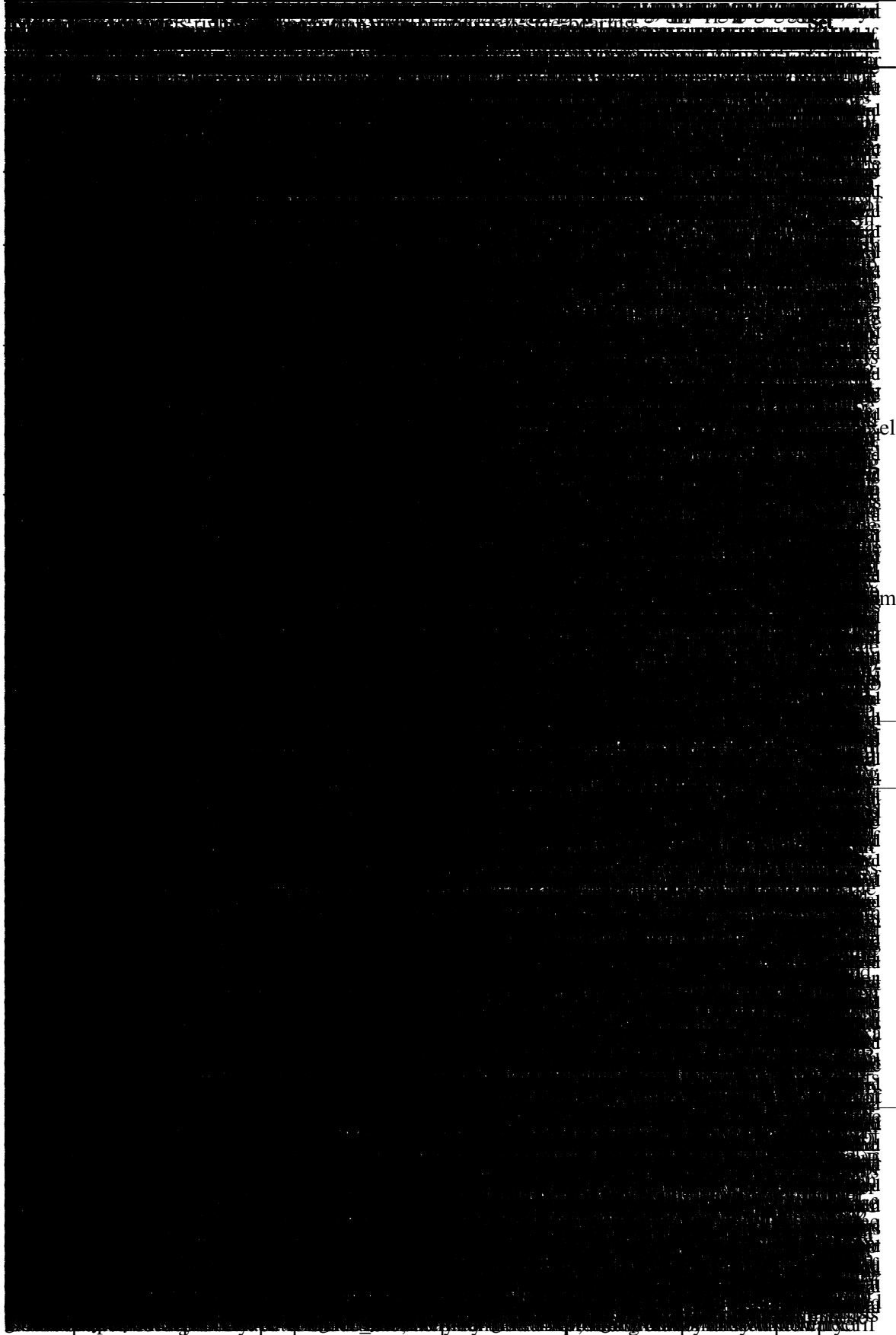
el
m-



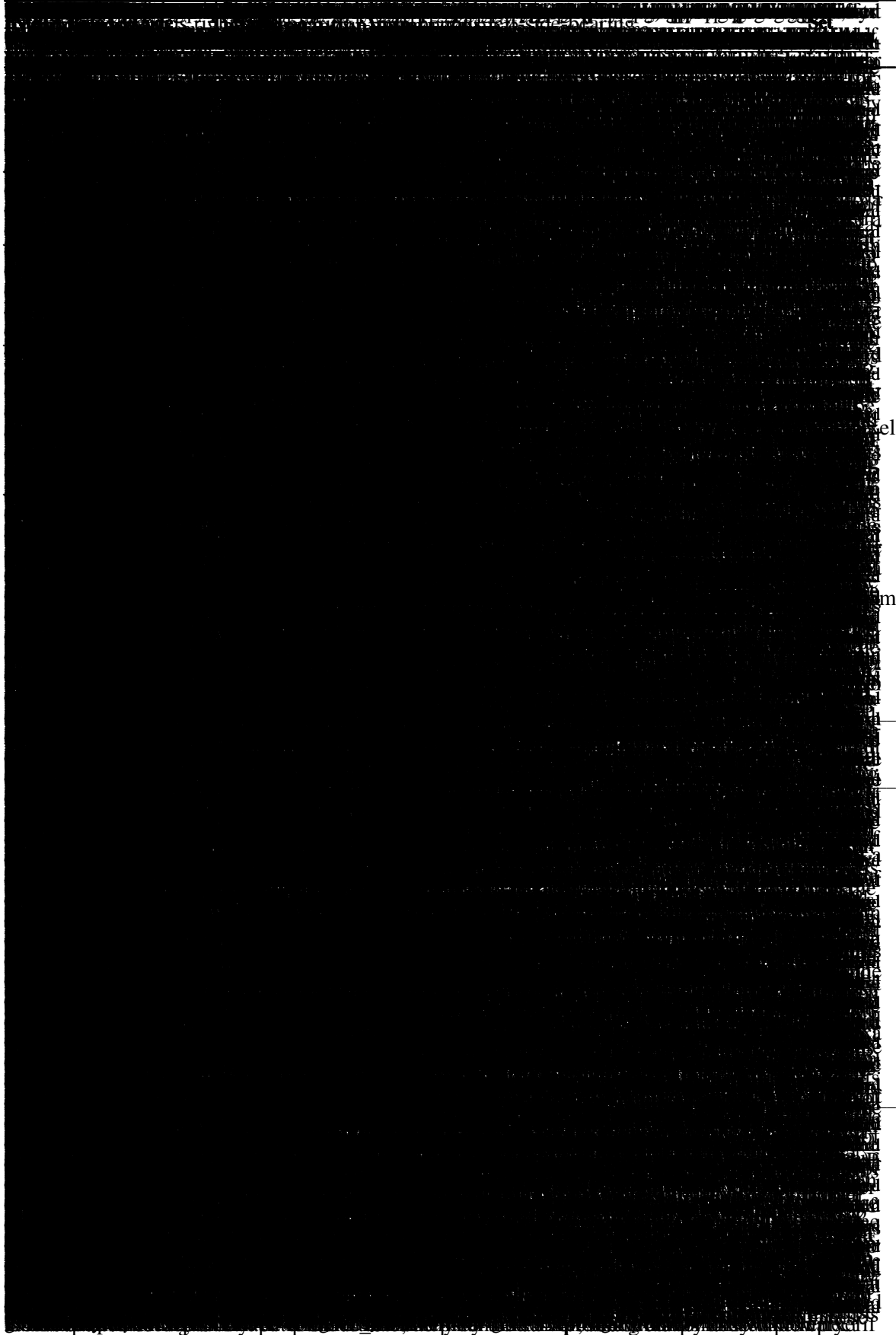
el
m-



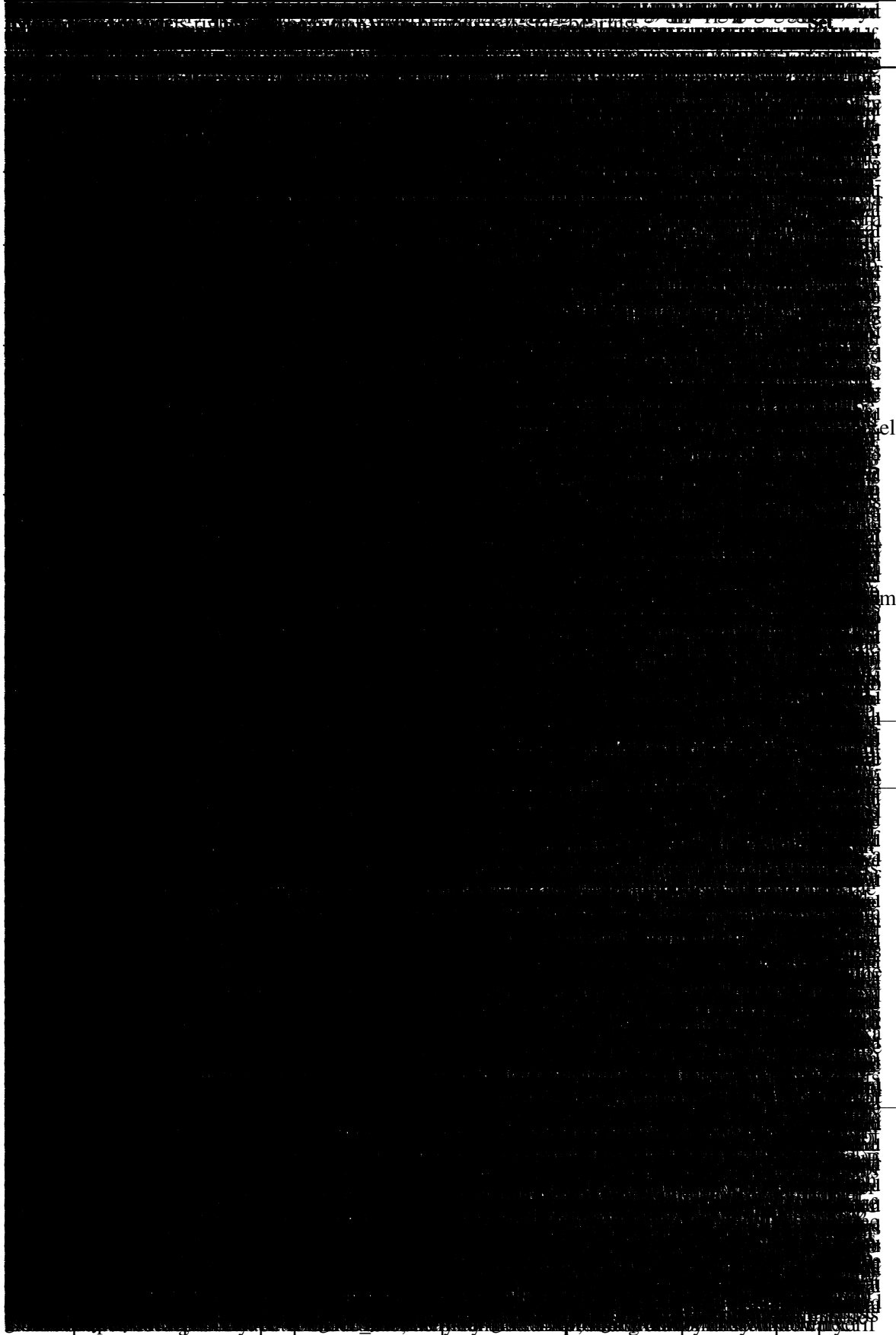
el
m-



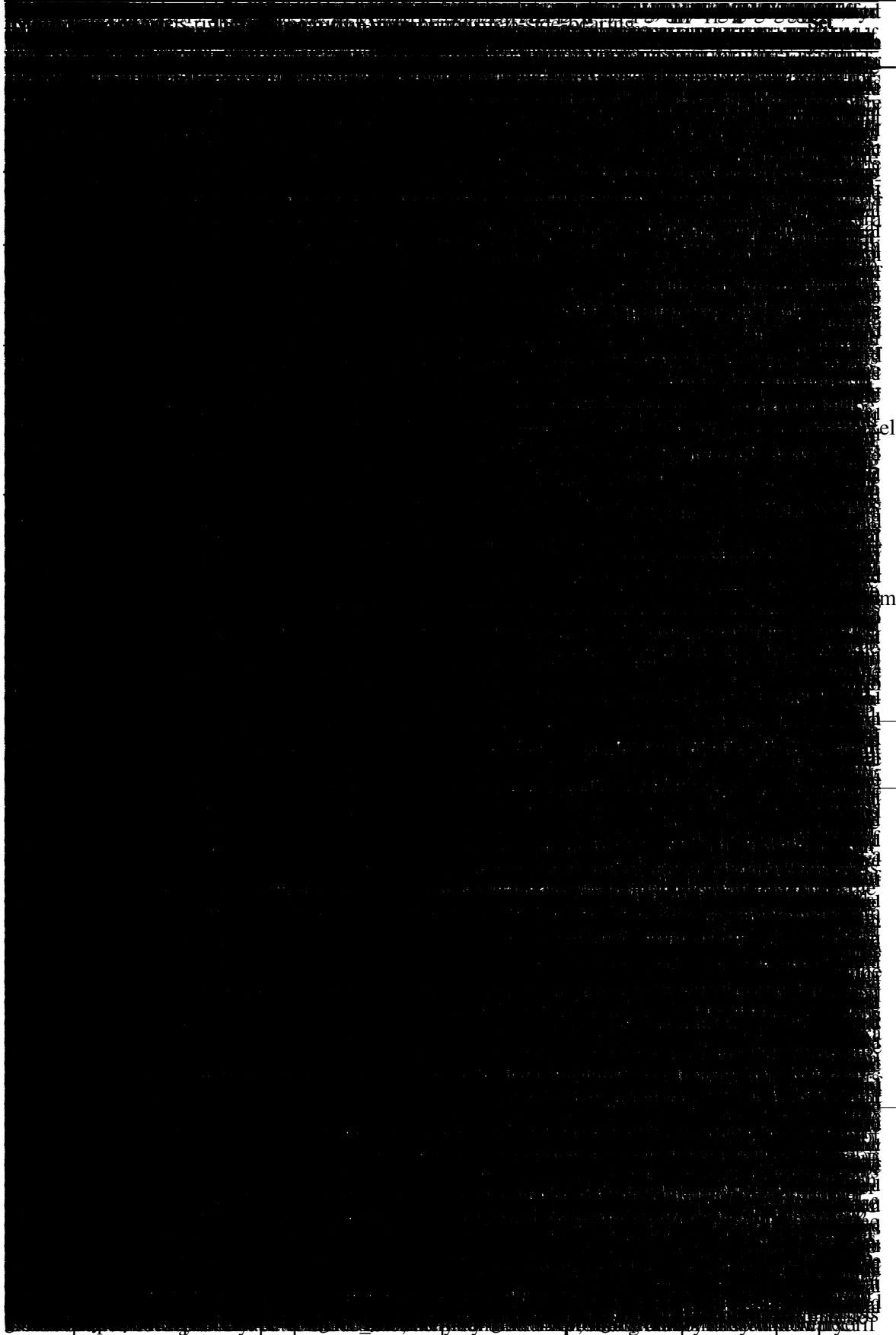
el
m-



el
m-

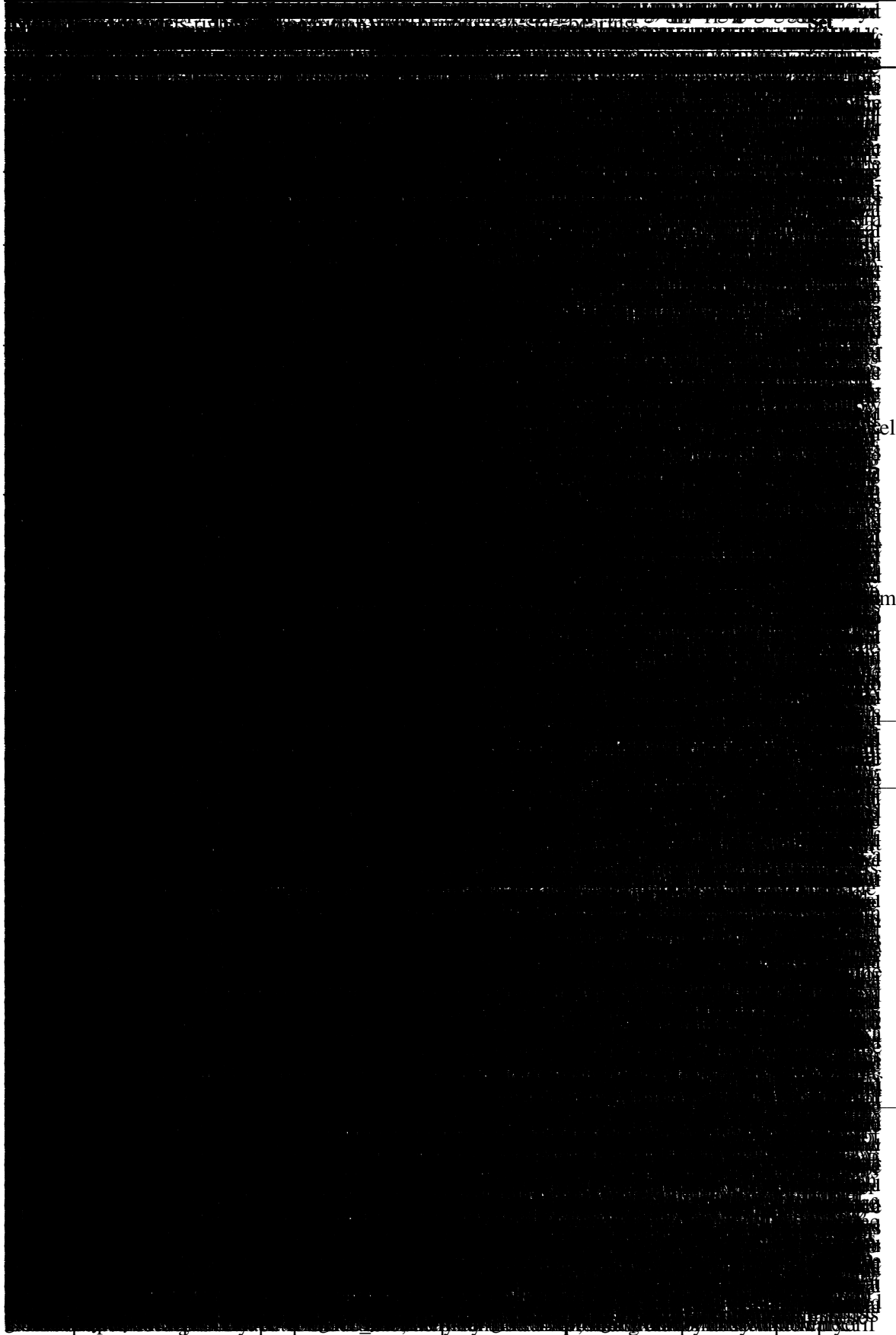


el
m-



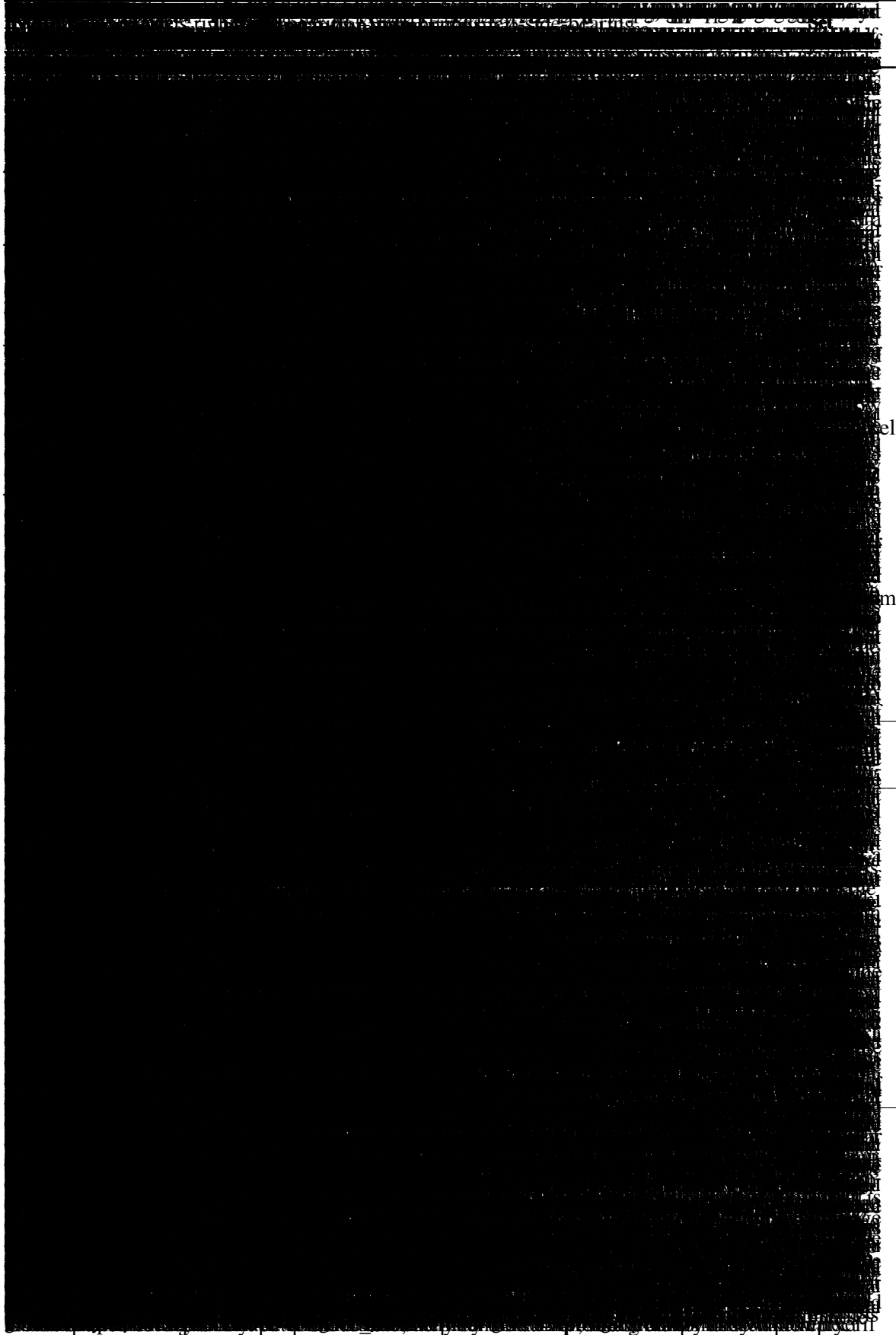
el

m-



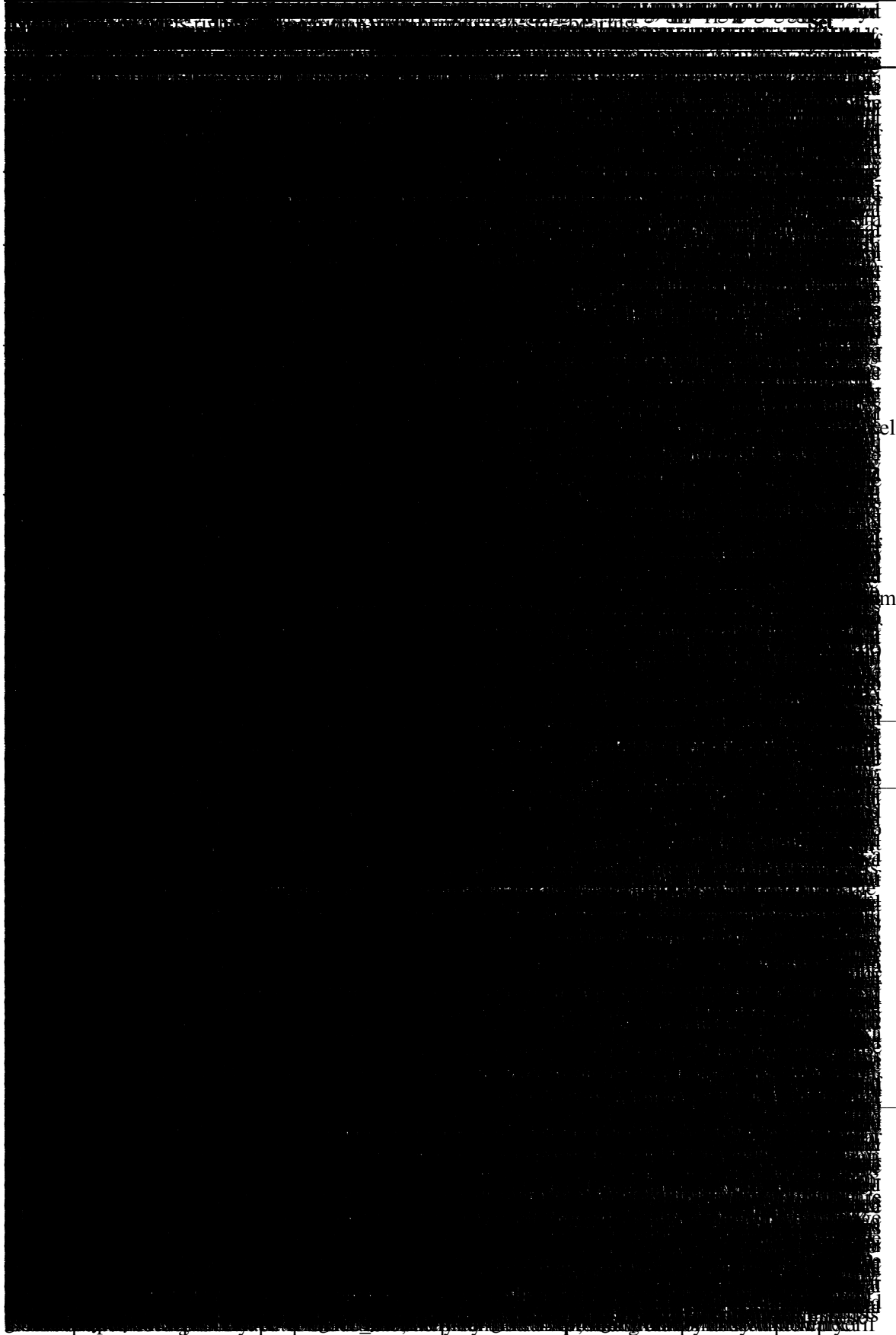
el

m-

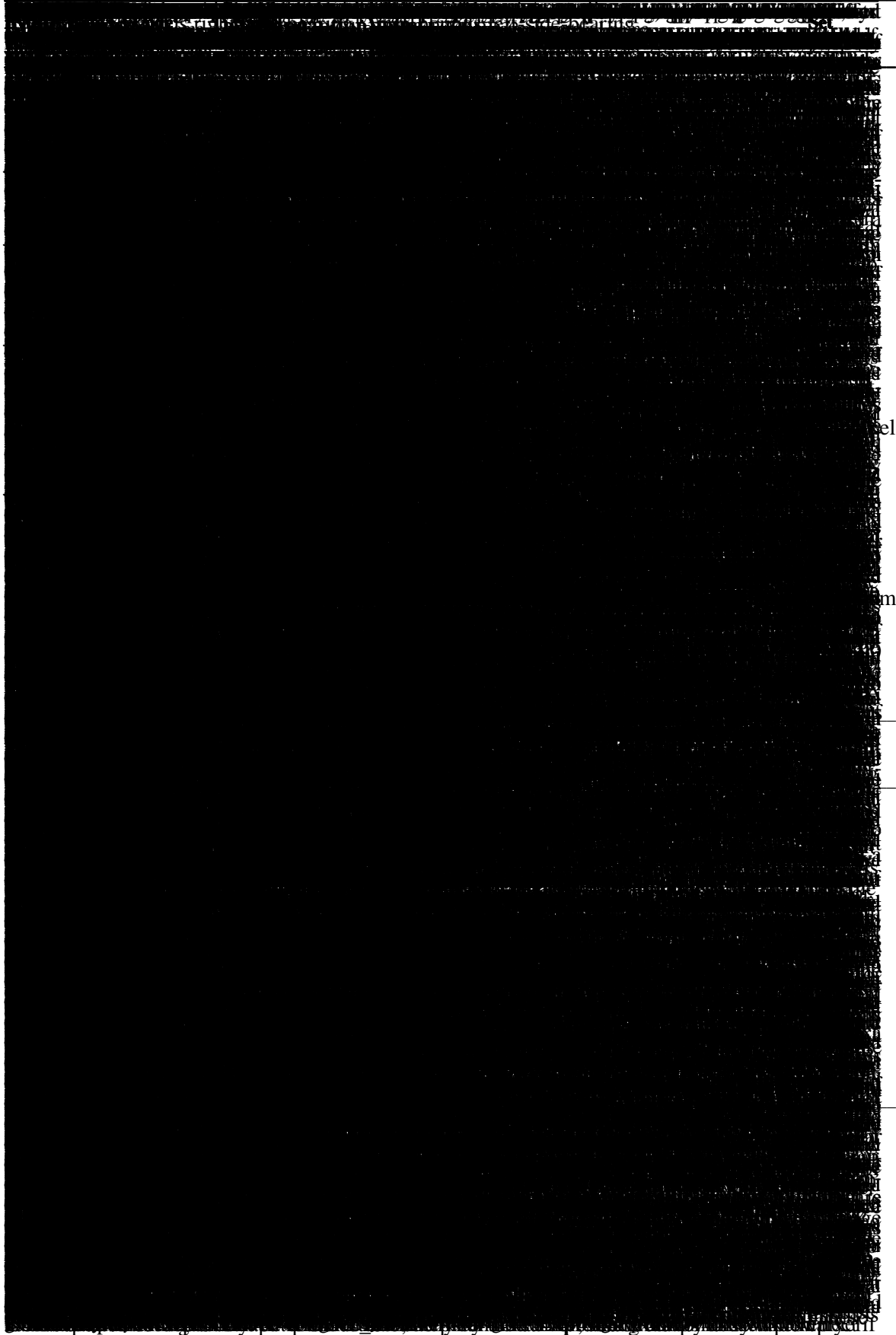


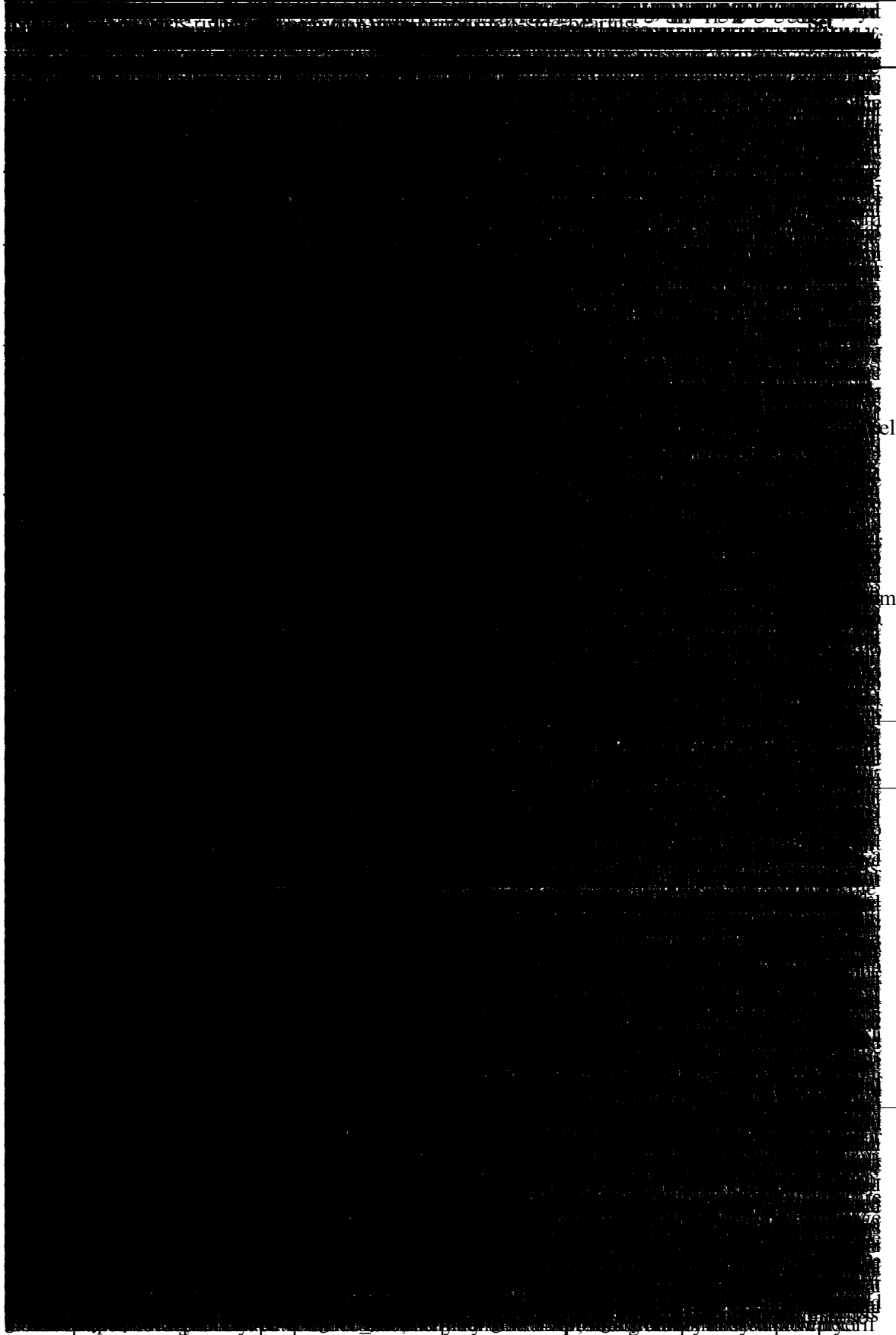
el

m-



sel
m-





el
m-