***COMPILER DESIGN***

***(10B17CI672)***

******

##### *PROJECT SYNOPSIS*

###### 

###### 

###### ***Submitted by:***

###### 

***VRISHTI GAHLAUT 12103528***

##### *SUMIT BANSAL 12103530*

##### *DHRUV SINGH 12103533*

##### *HAMZA KHAN 12103540*

**BATCH: B5TEACHER:MUKTA GOEL**

##### 

***In partial fulfillment for the award of the degree***

***Of***

***BACHELOR OF TECHNOLOGY***

***IN***

***COMPUTER SCIENCE AND ENGINEERING***

**ACKNOWLEDGEMENT**

We take this opportunity to express our profound sense of gratitude and respect to all those who helped us throughout the duration of this project. Preparing a project report is never a unilateral effort no matter ultimate credit may go to the author. We wish to acknowledge the guidance and support of the professors and academics in bringing up a real picture of the concept for which the report is prepared.

Our special thanks to our supervisor Ms. MUKTA GOEL who was so critical in this project and without her we would not have been able to do this.

We also thank all the staff members of JIIT for extending full support and making this whole experience enriching, informative and facilitating the project to reach its current state.

***JAVA TO C++ CONVERTER***

When writing Java applications, one of the more common things you will be required to produce is a parser. Parsers range from simple to complex and are used for everything from looking at command-line options to interpreting Java source code.

The purpose of lexical analyzers is to take a stream of input characters and decode them into higher level tokens that a parser can understand. Parsers consume the output of the lexical analyzer and operate by analyzing the sequence of tokens returned. The parser matches these sequences to an end state, which may be one of possibly many end states. The end states define the goals of the parser. When an end state is reached, the program using the parser does some action -- either setting up data structures or executing some action-specific code. Additionally, parsers can detect -- from the sequence of tokens that have been processed -- when no legal end state can be reached; at that point the parser identifies the current state as an error state. It is up to the application to decide what action to take when the parser identifies either an end state or an error state.

**DESIGNING INFORMATION**

**Platform: linux-ubuntu**

**Language Source: lex and yacc**

**Executable file : makefile**

**References**

* Wikipedia .
* Research paper related to project .
* documentation file and tutorials .
* Stack over flow problem discussions.
* conversion working .
* Youtube.com