Ose (Female)

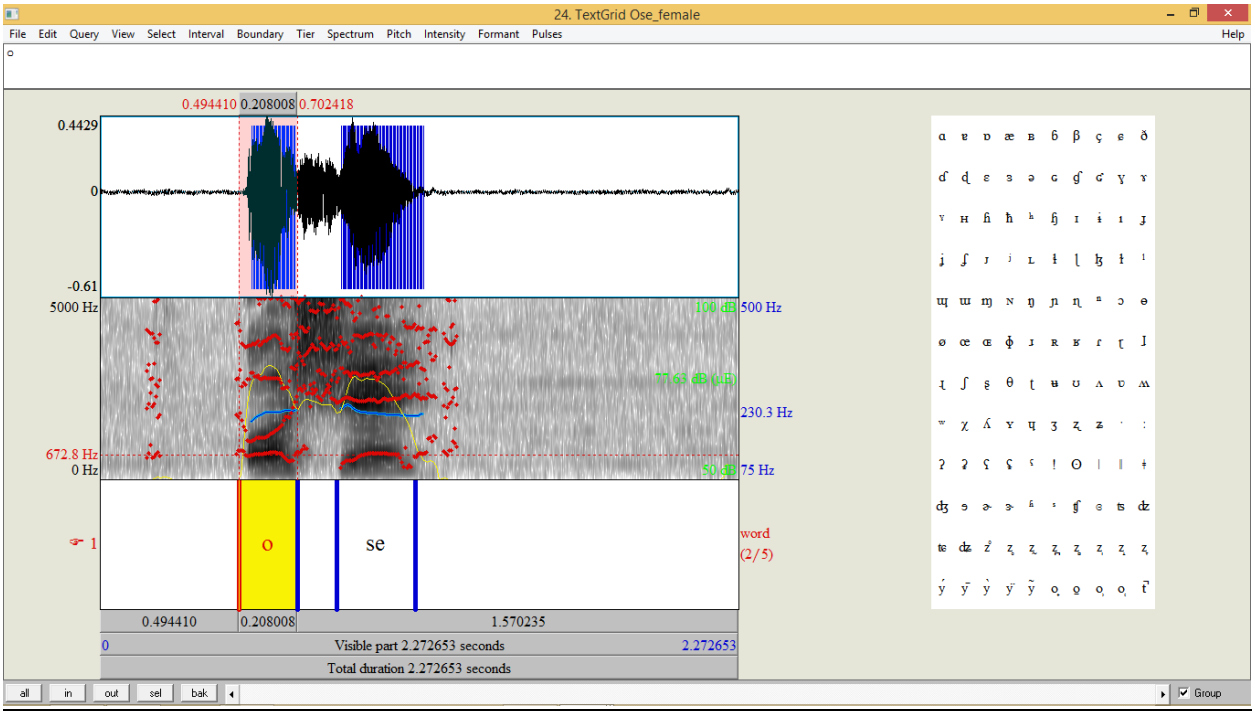

Figure 8.11 Analysis for Ose(Female)

F_0 for the first vowel– 1602Hz	F_0 for the second vowel – 227.5Hz
F_1 – 661.26Hz	
F_2 – 1391.61Hz	
F_3 – 2754.12Hz	
F_4 – 3770.78Hz	

Table 11: Analysis for Ose (Female)

Ose (Male)

F_0 for the first vowel– 616.5Hz	F_0 for the second vowel – 115.6Hz
F_1 – 598.97Hz	
F_2 – 1075.26Hz	
F_3 – 2861.77Hz	
F_4 – 3905.67Hz	

Table 12: Analysis for Ose (Female).

Procedures for Laboratory 9

Materials and tools

JLAP Software

Experiment Processes

Task 2

1. From the production rules:

   <S> : : = <A> <S> <B>

   S : → A → S →B →

   <A> : : → a

   A : → a

   <B> : : →b

B : → b

<S> : : λ

S : → λ

2. After the generation of G_0,

    a.

        i. The parse tree generated for a^3b^3 is:

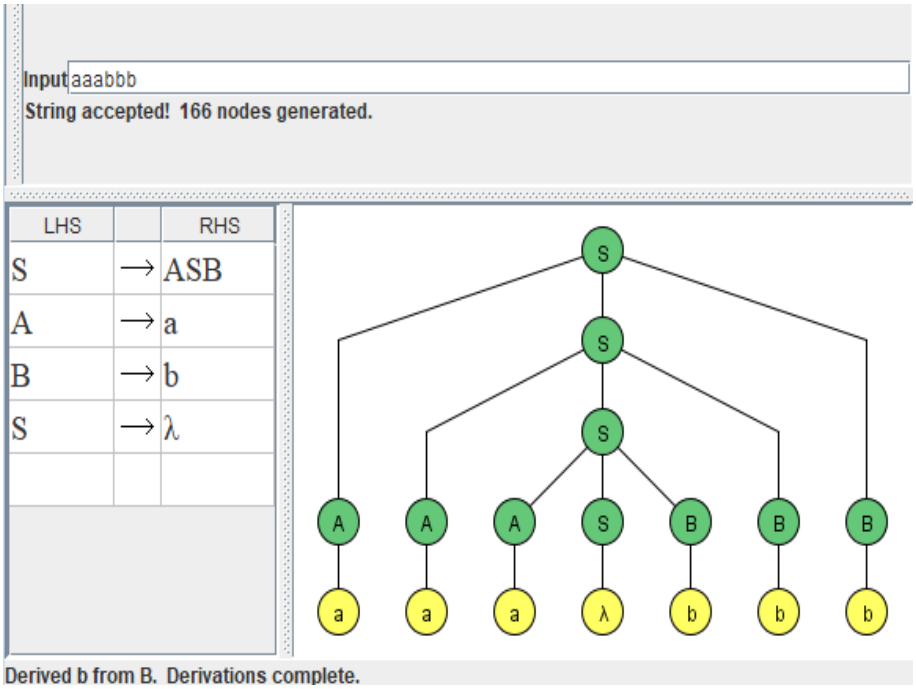

Figure 1: Parse tree for a^3b^3 using Grammar G_0.

ii. The parse tree generated for a^4b^4 is:

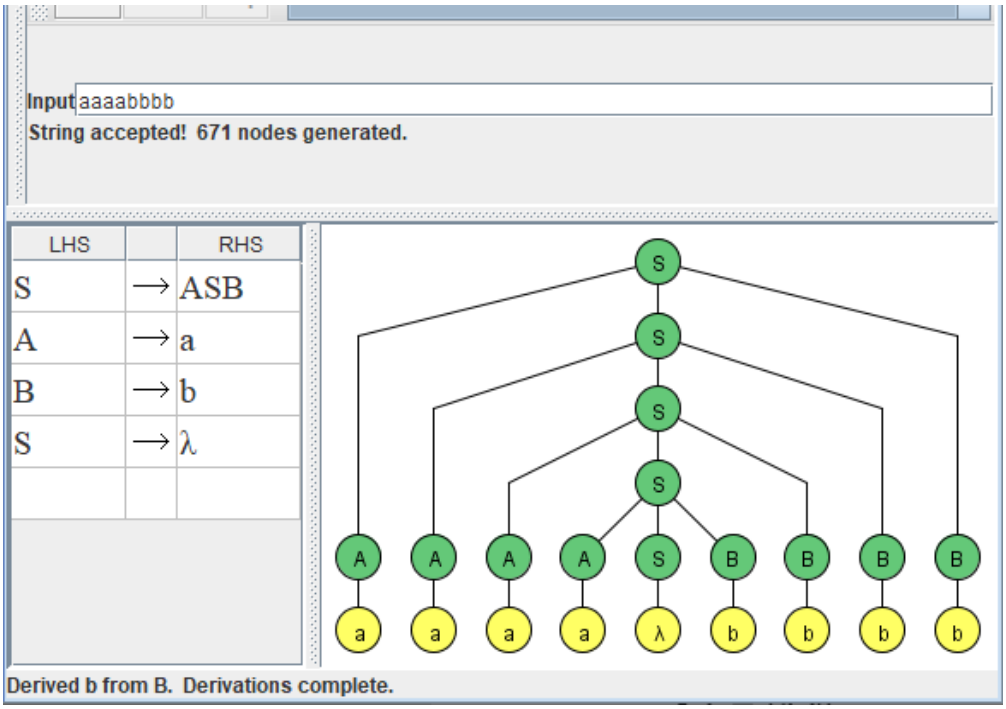

Figure 2: Parse tree for a^4b^4 using Grammar G_0.

iii. The parse tree generated for a^6b^6 is :

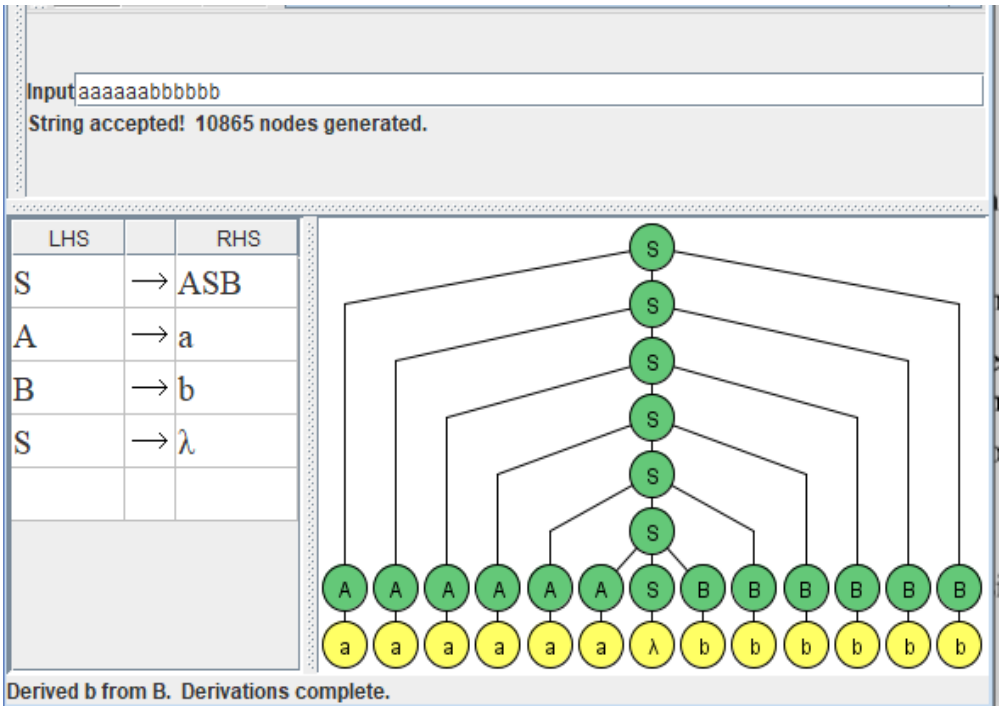

Figure 3: Parse tree for a^6b^6 using Grammar G_0.

b. For the expression of a^2b^3, the system generated

It generated error because a^2b^3 is not a valid string for the grammar.

Task 3

Grammar G_1 is a grammar that starts with the alphabet ‘a’ and ends with the alphabet ‘a’.

1. Railroad diagram

<S> : : = a <S> a

S : → a → S →a

<S> : : = b <S> a

S : → b → S →a

<S> : : λ

S : → λ

2. Parse tree for G_1

   All the inputs provided for Grammar G_1 did not generate a parse tree because they are invalid strings.

Grammar G_2 is a grammar that can be interpreted as a^nb^{2n} where $n \geq 1$.

1. Railroad diagram

<S> : : = a <S> bb

S : → a → S → b → b

<S> : : = abb

S : → a → b →b

2. Parse tree for G_2

   All the inputs provided for Grammar G_2 did not generate a parse tree because they are invalid strings.

Grammar G_3 is a grammar that starts with one or more ‘a’ and ends with lesser number of ‘b’ compared to the number of ‘a’.

All a’s in this grammar must be inputted before all the b’s.

1. Railroad diagram

   <S> : : = a <S>

   S : → a → S →

   <S> : : = <A>

   S : → A →

   <A> : : = ab

   A : → a → b

   <A> : : = a<A>b

   A : → a → A→ b

2. Parse tree generated for Grammar G_3

   i. The parse tree generated for a^3b^3 is :

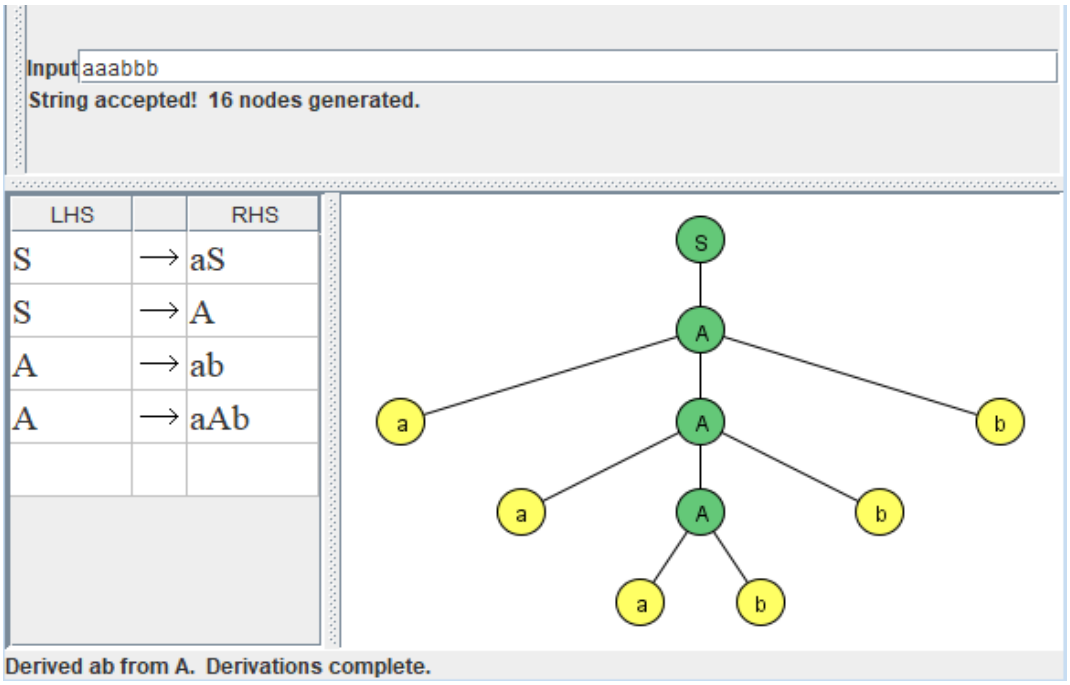

Figure 7: Parse tree generated for a^3b^3 using Grammar G_3.

ii. The parse tree generated for a^4b^4 is :

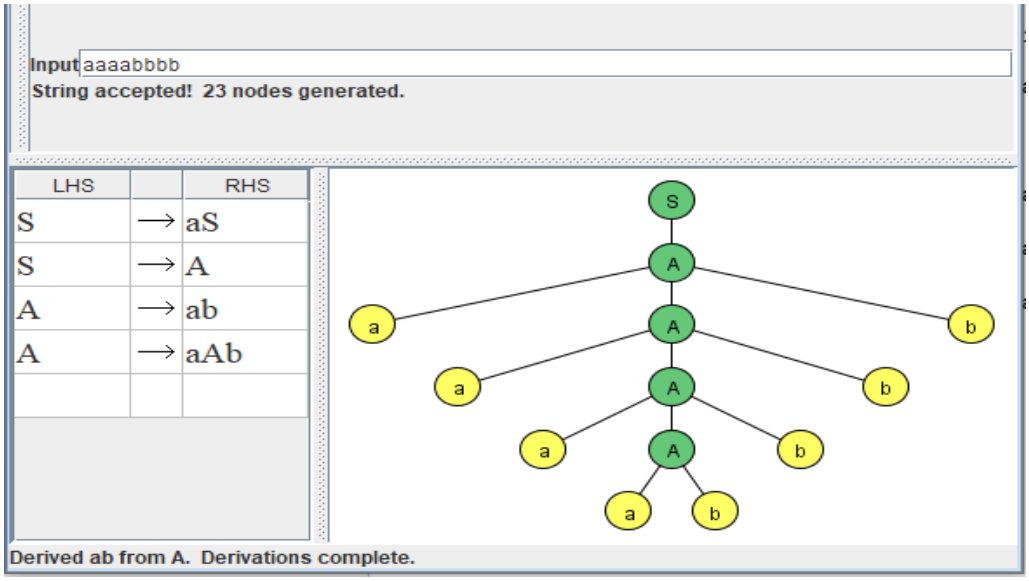

Figure 8: Parse tree generated for a^4b^4 using Grammar G_3.

iii. The parse tree generated for a^6b^6 is :

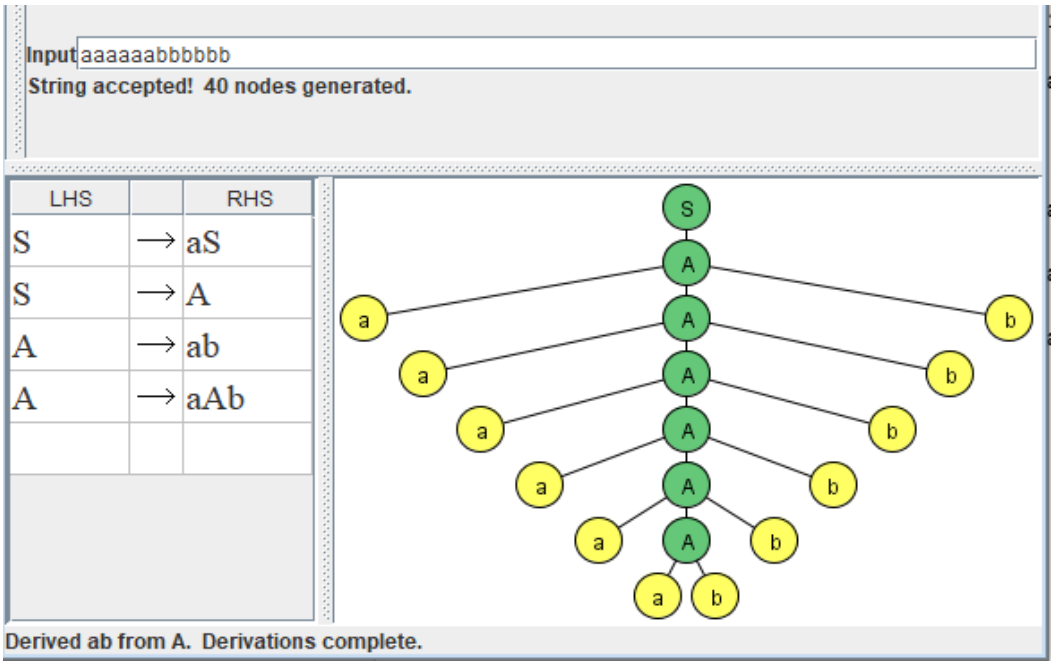

Figure 9: Parse tree generated for a^6b^6 using Grammar G_3.

iv. No parse tree was generated for a^2b^3 because it is an invalid string.

Grammar G_4 is a grammar that starts with one or more ‘a’ and ends with greater number of ‘b’ compared to the number of ‘a’.

All a’s in this grammar must be inputted before all the b’s.

1. Railroad diagram

   <S> : : = <S>b

   S : → S → b

   <S> : : = <B>b

   S : → B → b

   <B> : : = a<B>b

B : → a → B → b

<B> : : = ab

A : → a → b

2. Parse tree generated

For $a^n b^n$ inputs, no parse tree was generated because they are invalid strings.

Parse tree was generated for a^2b^3 because it is a valid input string.

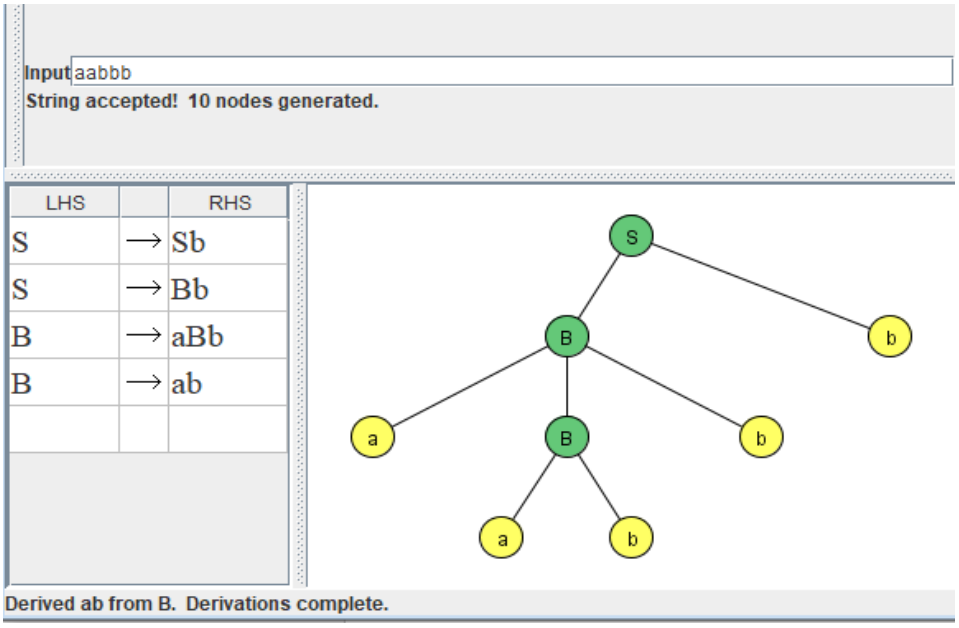

Figure 12: Parse tree generated for a^2b^3 using Grammar G_4.

Task 4

To design a grammar for base 6 number system, these must be defined:

$G_6 = <\Sigma, V, P, S>$

$\Sigma = \{0, 1, 2, 3, 4, 5\}$

$V = \{S, N, D\}$

P:

S → N

N → ND

N → D

D→ 0 |1 |2 |3 |4 |5

S is the string start symbol.

The valid strings are:

i. 543210

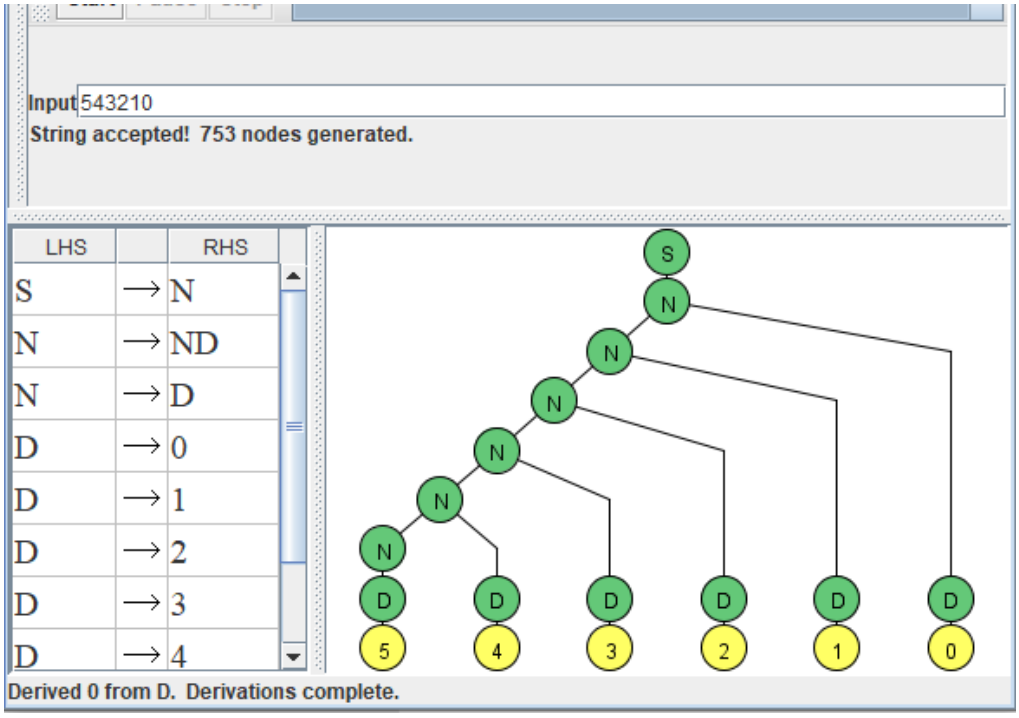

Figure 13: Parse tree for 543210 using Grammar G_6.

ii. 22

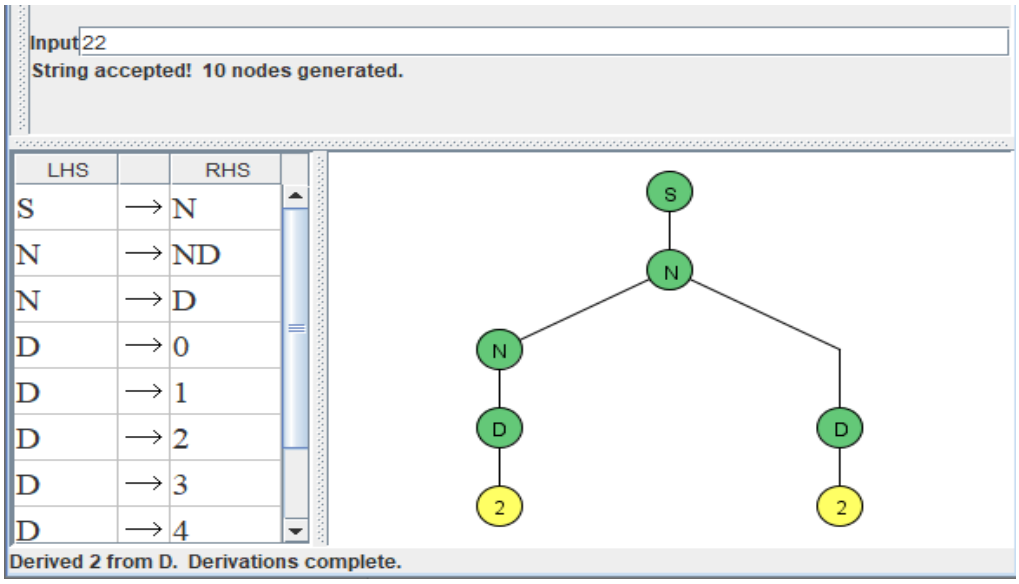

Figure 14: Parse tree for 22 using Grammar G_6.

iii. 043

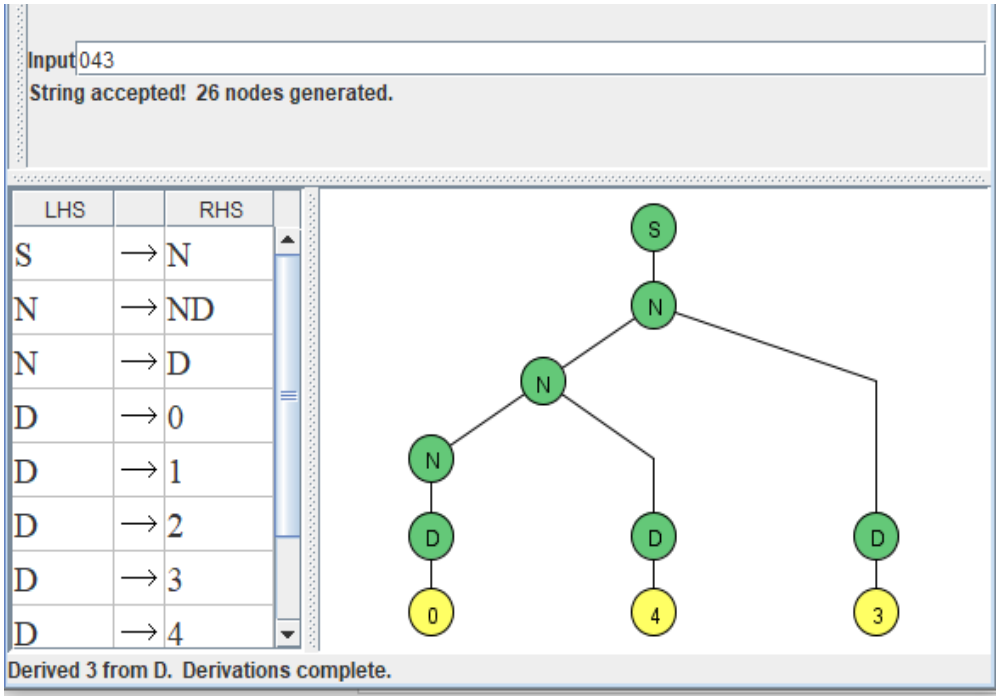

Figure 15: Parse tree for 043 using Grammar G_6.

iv. 12345

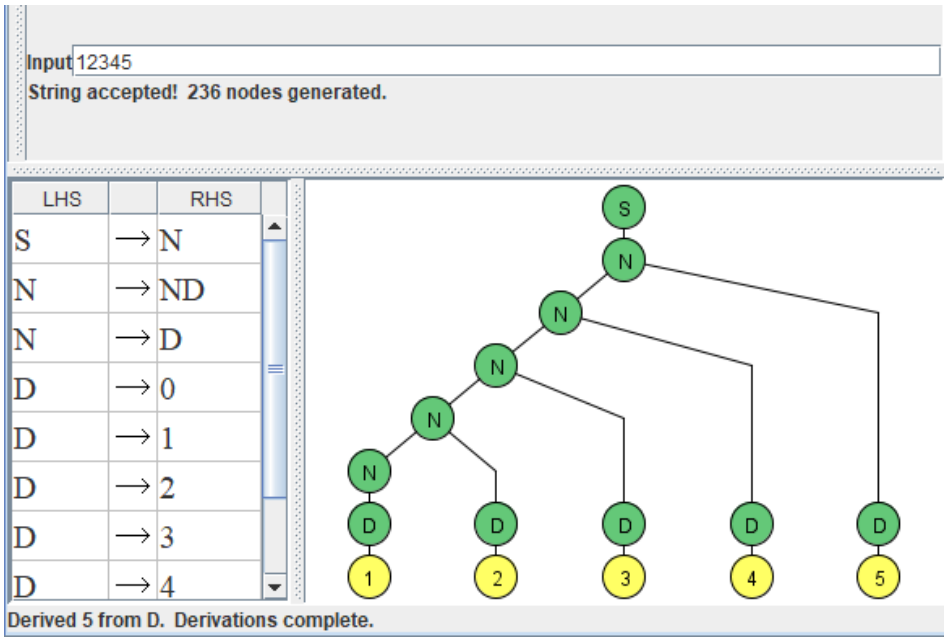

Figure 16: Parse tree for 12345 using Grammar G_6.

v. 4324

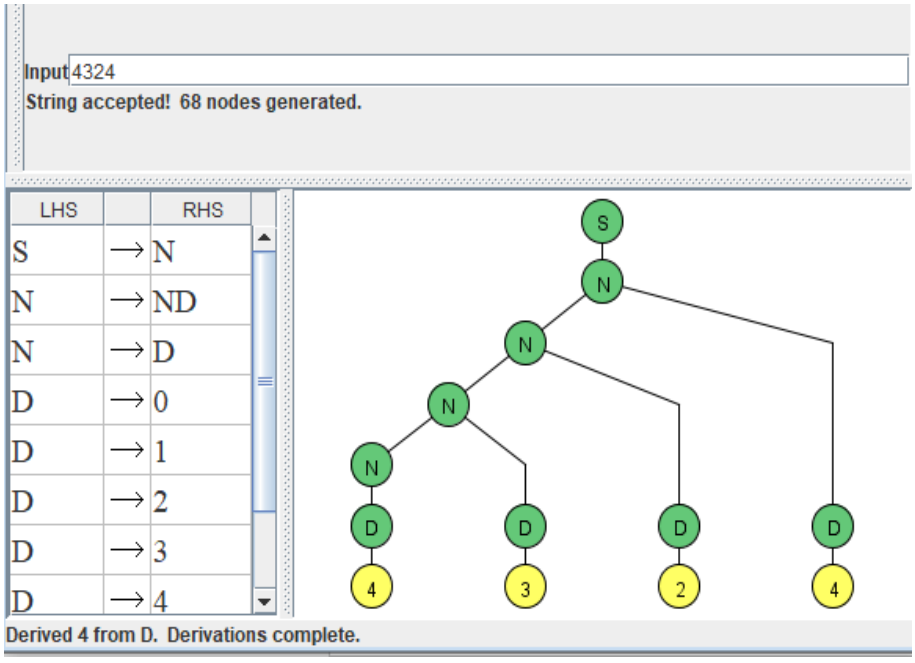

Figure 17: Parse tree for 4324 using Grammar G_6.

vi. 4

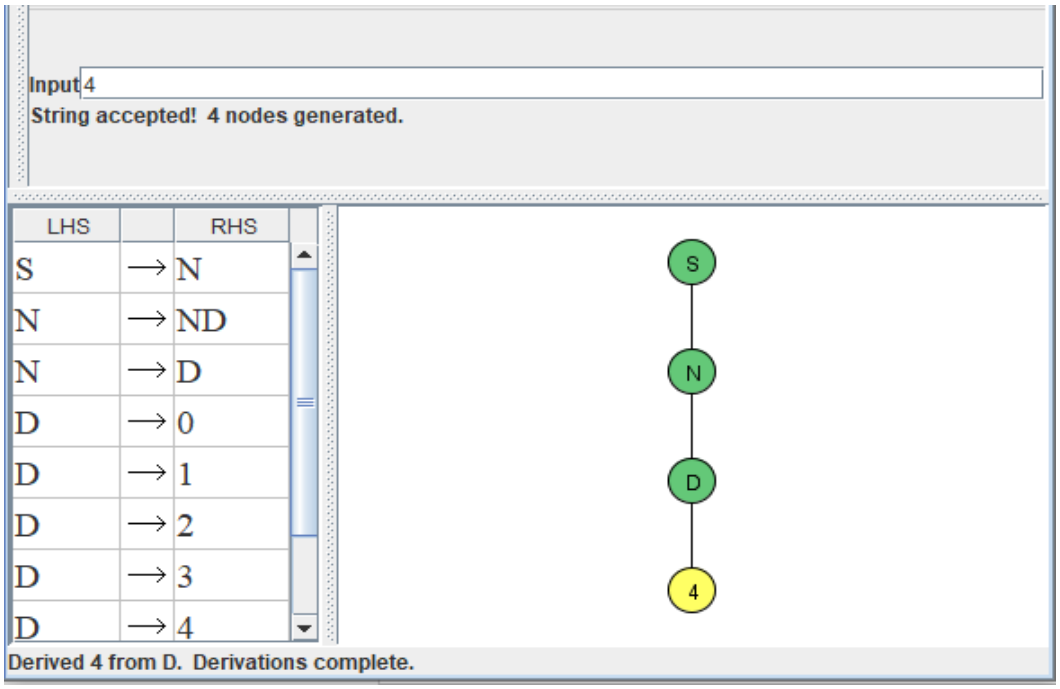

Figure 18: Parse tree for 4 using Grammar G_6.

The invalid strings are:

i. 8

ii. 69

iii. 990

iv. 6542

v. 73563

vi. 123456

The general output is:

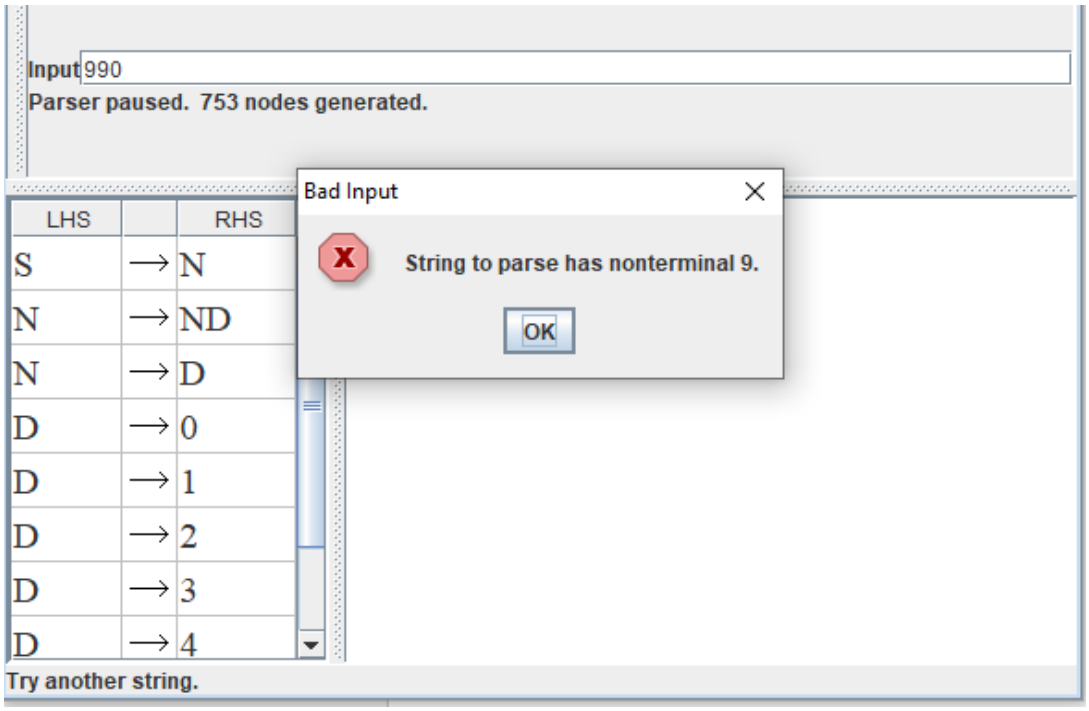

Figure 19: Invalid input for Grammar $G_{6.}$

The grammar is a context free grammar that accepts inputs from 0 to 5.

Procedures for Laboratory 10

Materials and tools

NLTK Library

TreeForm Software for drawing parse tree

Experiment Processes

Task 1

i. Olu go to the market

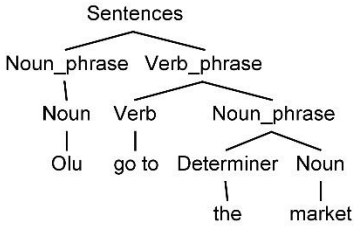

Figure 20: Parse tree for “Olu go to the market”.

ii. Ade sleep on the bed

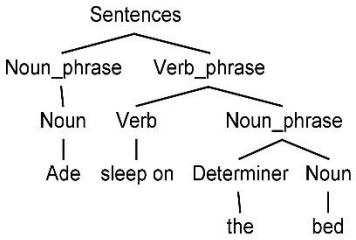

Figure 21: Parse tree for “Ade sleep on the bed”

iii. Shade take the book

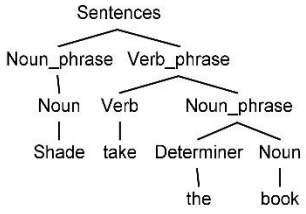

Figure 22: Parse tree for “Shade take the book”

iv. Kofo pick the pen

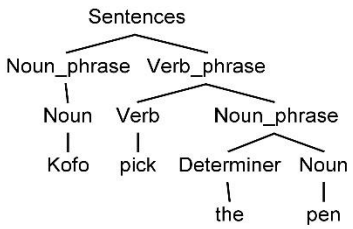

Figure 23: Parse tree for “Kofo pick the pen”

v. Francis cook the rice

vi. Yetunde wash the clothes

vii. Tomisin carry the bucket

viii. Ugo sweep the house

ix. Segun clean the bathroom

x. Wale lock the door

xi. Shukura bring the computer

xii. Bose enter the room

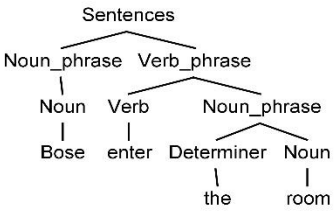

Figure 31: Parse tree for “Bose enter the room”

While exploring the English language correctness of the sentences generated using NLTK library, it was discovered that the grammar in the task didn’t account for preposition so it threw an error. Preposition was added to the grammar generated on the NLTK.

Preposition_phrase ::= preposition | preposition_phrase noun_phrase.

```
import nltk
grammar = nltk.CFG.fromstring("""
NP -> D NP | N
VP -> V NP | V
PP -> P | PP NP
P -> 'to' | 'on'
D -> 'the'
N -> 'Olu' | 'market' | 'Ade' | 'bed' | 'Shade' | 'book' | 'Kofo' | 'pen'
N -> 'Francis' | 'rice' | 'Yetunde' | 'clothes' | 'Tomisin' | 'bucket' | 'comput
N -> 'Ugo' | 'house' | 'Segun' | 'bathroom' | 'Wale' | 'door' | 'Shukura'
V -> 'go' | 'sleep' | 'take' | 'pick' | 'cook' | 'wash' | 'carry' | 'sweep'
V -> 'clean' | 'lock' | 'bring' | 'enter'
""")

sentence = 'Tomisin carry the bucket'.split()
parser = nltk.ShiftReduceParser(grammar, trace = 2)
for tree in parser.parse(sentence):
    print(tree)
```

Figure 32: Code to ascertain the correctness of English Language grammar using NLTK.

```
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================ RESTART: C:/Users/KOFOWOROLA/Desktop/Nltk.py ================
Warning: VP -> V NP will never be used
Parsing 'Tomisin carry the bucket'
    [ * Tomisin carry the bucket]
  S [ 'Tomisin' * carry the bucket]
  R [ N * carry the bucket]
  R [ NP * carry the bucket]
  S [ NP 'carry' * the bucket]
  R [ NP V * the bucket]
  R [ NP VP * the bucket]
  S [ NP VP 'the' * bucket]
  R [ NP VP D * bucket]
  S [ NP VP D 'bucket' * ]
  R [ NP VP D N * ]
  R [ NP VP D NP * ]
  R [ NP VP NP * ]
>>>
```

Figure 33: Result when tested with an English sentence

Task 2

1. The Yoruba Language grammar is as follows:

   <sentence> ::= <noun phrase><verb phrase>

   <noun phrase> ::= <proper noun> | <noun><determiner>

   <verb phrase> ::= <verb> | <verb><noun phrase>

   <noun> ::= {List all noun}

   <verb> ::= {List all verb}

   <determiner> ::= {List all determiner}

2. Sentences and parse trees

   i. Olu lo si oja

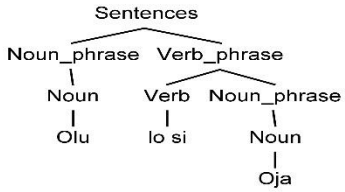

Figure 34: Parse tree for “Olu lo si oja”

ii. Ade sun sori ibusun

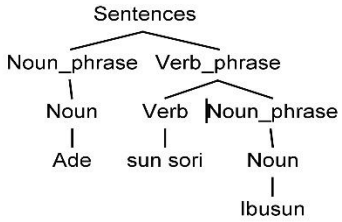

Figure 35: Parse tree for “Ade sun sori ibusun”

iii. Shade gba iwe naa

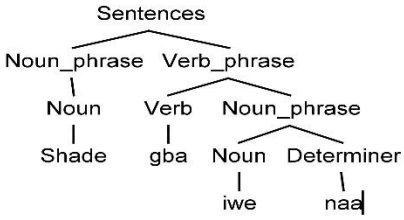

Figure 36: Parse tree for “Shade gba iwe naa”

iv. Kofo mu peni

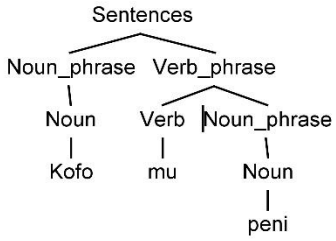

Figure 37: Parse tree for “Kofo mu peni”

v. Francis se iresi

vi. Yetunde fo aso

vii. Tomisin gbe koroba

viii. Ugo gba inu ile

ix. Segun fo baluwe

x. Wale ti ilekun

xi. Shukura gbe computer

xii. Bose wo inu yara

```
import nltk
grammar = nltk.CFG.fromstring("""
NP -> N D | N
VP -> V NP | V
PP -> P | PP NP
P -> 'si' | 'sori'
D -> 'naa'
N -> 'Olu' | 'oja' | 'Ade' | 'ibusun' | 'Shade' | 'iwe' | 'Kofo' | 'peni'
N -> 'Francis' | 'iresi' | 'Yetunde' | 'aso' | 'Tomisin' | 'koroba' | 'komputa'
N -> 'Ugo' | 'ile' | 'Segun' | 'baluwe' | 'Wale' | 'ilekun' | 'Shukura'
V -> 'lo' | 'sun' | 'gba' | 'mu' | 'se' | 'fo' | 'gbe' | 'gba'
V -> 'fo' | 'ti' | 'gbe' | 'wo'
""")
sentence = 'Tomisin gbe koroba naa'.split()
parser = nltk.ShiftReduceParser(grammar, trace = 2)
for tree in parser.parse(sentence):
    print(tree)
```

Figure 38: Code to ascertain the correctness of Yoruba Language grammar using NLTK

```
================ RESTART: C:/Users/KOFOWOROLA/Desktop/Nltk.py ==============
Warning: NP -> N D will never be used
Warning: VP -> V NP will never be used
Warning: V -> 'gba' will never be used
Warning: V -> 'fo' will never be used
Warning: V -> 'gbe' will never be used
Parsing 'Tomisin gbe koroba naa'
    [ * Tomisin gbe koroba naa]
  S [ 'Tomisin' * gbe koroba naa]
  R [ N * gbe koroba naa]
  R [ NP * gbe koroba naa]
  S [ NP 'gbe' * koroba naa]
  R [ NP V * koroba naa]
  R [ NP VP * koroba naa]
  S [ NP VP 'koroba' * naa]
  R [ NP VP N * naa]
  R [ NP VP NP * naa]
  S [ NP VP NP 'naa' * ]
  R [ NP VP NP D * ]
>>>
```

Figure 39: Result when tested with a Yoruba sentence

3. A situation of ambiguity arises when the determinants in the Yoruba language do not appear at all or appear at the end of the sentences. This could be curbs by adding them automatically during translation.

Ambiguity also arises when two or more English verbs can translate into only one Yoruba verb. This can be curbed by translating according to the context of the sentence.

4. During the translation process, the nouns are easily understood and duly translated. The verbs on the other hand are very difficult to translate without knowing the context of the whole sentence.

Task 3

1. The development of a software that checks the correctness of English statements based on the grammars defined above can be inferred from Figure 32 and 33
2. The development of a software that checks the correctness of Yoruba statements based on the grammars defined above can be inferred from Figure 38 and 39.
3. The system failed to properly compile without adding prepositions to the grammar. For all other sentences, the system worked properly which can be inferred from Figure 32, 33, 38 and 39.
4. The translation development software worked for both the English and Yoruba grammar which resulted in similar grammatical translations except for the fact that in the Yoruba translation grammar, the Determiner, if there is any is always at the end of the sentence. This could be inferred from Figure 39.

Procedures for laboratory 11

Materials and tools

Python programming language

Experiment Processes

Limitation of the System

1. The major limitation of the is that the system did not give room for spell checking, therefore little mistake results in grievous errors
2. The system did not have autocomplete nor any hint to aid communication
3. The word list is not visible for user
4. The error messages are not explanatory
5. The interaction is not spontaneous.

Description of python programming language

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted:** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive:** You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

- **Python is a Beginner's Language:** Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, Unix shell, and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python's features include:

- **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read:** Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain:** Python's source code is fairly easy-to-maintain.
- **A broad standard library:** Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

- **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases:** Python provides interfaces to all major commercial databases.
- **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable:** Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

DISCUSSION OF RESULTS

The result of the tasks in each laboratory was appropriately discussed during the experiment processes.

TECHNOLOGICAL INTERPRETATION AND APPLICATION OF RESULTS

Using the results from laboratory 10, a real life Yoruba Language grammar generator could be implemented appropriately when all errors and observations are taken into consideration in order to foster a better Yoruba translation system.

For a novice, a real life development of robotic arm could be implemented for educational and learning purposes while following the directives of this report.

Using results from laboratory 9, valid grammar can now be generated efficiently following the directives of this report.

CONCLUSION AND SUGGESTIONS FOR FURTHER WORKS

All the laboratory tasks in this project have been attempted by all the members in this group, though not every task could be completed based on the limited knowledge of individuals in this project group.

The next course of action is to continue developing solutions to tasks we are not satisfied with and improving in the areas we lack as regards knowledge of some particular tasks.

REFERENCES

https://www.everyculture.com/wc/Mauritania-to-Nigeria/Yoruba.html

https://www.britannica.com/topic/Yoruba

Bamgbọṣe, Ayọ (1965). *Yoruba Orthography*. Ibadan: Ibadan University Press