IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 21, Issue 6, Ser. I (Nov - Dec 2019), PP 08-15
www.iosrjournals.org

Comparative Analysis of Improving Computer Programming Skills of Information Technology First Year Students at Ambo University - An Action Research

BayisaNateaSima[#1], Sasikumar Perumal[#2]
[#1]*Lecturer and Head, Information Technology, Ambo University, Ethiopia*
[#2]*Assistant Professor, Information Technology, Ambo University, Ethiopia*

Abstract
The teaching of computer programming is one of the greatest challenges that have remained for years in Information Technology department.Teaching programming is one of the foundations of Information Technology education.One of the major issues related to teaching computer programming course is the excessiveamount of time spent on the understanding problem, writing algorithms, coding, and language's syntax, which leaves little time for developing skills in program design and solution creativity. The wide variation in the students' backgrounds, coupled with the traditional classroom teaching strategy, and bounded course duration, makes it extremely difficult for an instructor to go beyond adequate syntax coverage, to developing and enhancing the student's problem-solving abilities. The solution to this problem is facilitating a teaching environment that transforms and enhances traditional classroom teaching with customized software teaching tool.The aim of this paper is how to increase student programming skillssuccessful in the optimization of teaching computer programming for beginners. Here we are introducing pre- intervention test and post-intervention test to check the improvement/enhancement of the students during intervention.The Action Research seeks to discover ways to improve thecomputer programming for regular second year Information Technology in ambo university, Ethiopia. The goal is to help the beginners develop their programming skills, proffer a teaching technology that maximizes students' chances of engagement and help them to become learners of programming.Additionally, beginners will be able to operate the computer, program, and improve their programming skills through involvement. Perhaps one of the first challenges is tackling and changing or improving the places within which many of us practice.

Date of Submission: 16-11-2019 Date of Acceptance: 30-11-2019

I. Introduction

Learning is the process by learners acquires knowledge, develop skills and bring about attitudinal changes because of the interactions they establish among themselves and with their teachers.In recent years the demand for programmers and student interest in programming has grown rapidly and introductory programming courses have become increasingly popular.Programming is a very useful skill and learn to program is hard however. Programming courses are generally regarded as difficult, and often have the highest dropout rates.When students join University, they are very low in computer programming. This problem may from student attitude towards programming (like programming is difficult and time consuming to practice), low problem solving skill, difficult to understand algorithms, design program, steps to solve problem, unable to understand programming language (like C, C++, and java), syntax features (like editing, compile, linking, and execute program) less practice in lab, and less skill of trainers. Jenkins,T. (2002).

Difficult to learn, programming skills are difficult to teach too (Allison, Orton & Powell, 2002), not least because "traditional teaching methods do not adapt well to the domains of coding and problem solving, as it is a skill best learned through experience" (Traynor& Gibson, 2004, p. 2).

According to Kölling and Rosenberg (2001), the situation is even more challenging when it comes to teaching object-oriented programming to beginning students as "software tools, teaching support material and teachers' experience all are less mature than the equivalent for structured programming". Here, too, students struggle with programming, and programming has continued to be a major factor contributing to the attrition of first year students from the computing courses. To address the difficulties associated with computer programming, first it is necessary to understand them well. All the university follows traditional way of teaching (lecturing) computer programing for those students enrolls for first year in university).

Programming languages typically used in programming classes are professional in nature, such as C, C++, C# and Java; they have extensive and complex syntaxes, rendering learning difficult for beginners (Jenkins, 2002; Motil& Epstein, 2000*). Students' difficulties with abstract concepts*-knowing howto design a

solution to a problem, subdivide it into simpler code able subcomponents, and conceive hypothetical error situations for testing and finding out mistakes (Morgado&Martins, 2008); difficulties in understanding even the most basic concepts (Miliszewska& Tan, 2007) such as variables, data types or memory addresses as these abstract concepts do not have direct analogies in real life (Miliszewska& Tan); and not knowing how to use the programming language correctly to create a program (Winslow, 1996).

According to the observation of the author, teaching computer programming is one of the foundations of Information technology education. It is important for the teachers to gain students' attention and strengthen their motivation for learning to program with the help of a variety of teaching methods. Learning programming is a difficult task for many students and teachers are looking for methods to improve this.

1.1. Problem of students in learning programming

A wide variety of difficulties students encountered during programming were listed below.
These are classified as follows, mainly based on the programming process aspect (Shefali, 2016)

I. Starting problems
II. Language syntax related
III. Logic related
IV. Debugging related
V. Lack of knowledge related to various aspects like OS, problem domain etc.

The followingsdescriptions are based on the classification of educational objective of bloom taxonomy of Cognitive domain and Associated programming difficulties.

1. **Knowledge**: Recall data
   - Difficulty to remember the syntax of a Programming Language and makes syntax errors.
   - Lack of knowledge of useful library functions and header files.
   - Lack of knowledge about system software.
   - Forget to declare/initialize variables.
   - Lack of problem-domain knowledge.
2. **Comprehension:** Understand the meaning, translation, interpolation, and interpretation of instructions and problems; state a problem in one's own words.
   - Difficulty to understand the problem to solve
   - Unable to interpret the compiler generated error/warning messages
   - Difficulty in translating a logic to program
   - Difficulty in minimizing the number of steps
   - Difficulty to comprehend a given program
3. **Application**: Use a concept in a new situationor use an abstraction unprompted; apply what was learned in the classroom to novelsituations in the workplace.
   - Unable to apply theoretical knowledge
   - Unable to solve a given problem
   - Difficulty in algorithm design
   - Difficulty in code optimization
4. **Analysis**: Separate material or concepts intocomponent parts so that its organizational structure may be understood; distinguishbetween facts and inferences.
   - Difficulty to comprehend a program without comments
   - Difficulty to understand recursive functions
   - Difficulty to understand the logic of large/complex programs
5. **Synthesis**: Build a structure or pattern from diverse elements; put parts together to form a whole structure, with emphasis on creating a new meaning or structure.
   - Taking more time to find a correct logic
   - Difficulty in algorithm design
   - Difficulty to integrate different modules
6. **Evaluation**: Make judgments about the valueof ideas or materials.
   - Unable to describe a program logic
   - Difficulty to justify and debug a program logic
   - Taking more time to debug a program

1.2. How Do We Write a Program?

A computer is not intelligent. It cannot analyze a problem and come up with a solution.A human (the *programmer*) must analyze the problem, develop the instructions for solving the problem, and then have the computer carry out the instructions. To write a program we should follows the following steps carefully.

1.2.1. Understanding(define)problem

Before we are going to solve problems and develop steps, we could understand problem (i. e the central idea, core point, reasoning, associate with real problems andcompare with real-life, logical thinking and imagine the problems first). Logical thinking is an important foundation skill.Albrecht says that the basis of all logical thinking is sequential thought. This process involves taking the important ideas, facts, and conclusions involved in a problem and arranging them in a chain-like progression that takes on a meaning in and of itself. To think logically is to think in steps.

Programming is a process of problem solving. To be a good problem solver and a good programmer, you must follow good problem-solvingtechniques. One common problem-solving technique includes analyzing a problem, outlining the problem requirements, and designing steps, called an algorithm, to solve the problem (D. S. Malik). A clearly defined problem is already half the solution.

For example, **Problem one**

Write a program that will take as input the type of restaurant the user ate at, the cost of the meal, list of meal, the number of people in his/her party, and how good the service was.Determine the dollar amountof the tip:

Therefore, during problem understanding students could identify input variable, process variable, and output variable clearly. At the same time students could take into account, logical steps and develop steps to solve problems without any ambiguity.

II. Literature Review

Computer programming is a fundamental skill that all Information Technology students are required to learn.However, programming courses are generally regarded as difficult and often have the highestdropout rates (Gomes, Areias, Henriques& Mendes, 2008 ;). In the scientific literature, many reasons are pointed out for this, such as the following.

Methodology and tools used—traditional teaching methods, normally based on lectures and specific programming language syntaxes, often fail in what concerns the students' motivation in getting involved in meaningful programming activities (Schulte &Bennedsen, 2006).

Programming languages typically used in programming classes are professional in nature, such as C, C++, C# and Java; they have extensive and complex syntaxes, rendering learning difficult for beginners (Jenkins, 2002; Motil& Epstein, 2000). *Students' difficulties with abstract concepts*—knowing howto design a solution to a problem, subdivide it into simpler code able subcomponents, and conceive hypothetical error situations for testing and finding out mistakes (Morgado&Martins, 2008); difficulties in understanding even the most basic concepts (Miliszewska& Tan, 2007) such as variables, data types or memory addresses as these abstract concepts do not have direct analogies in real life (Miliszewska& Tan); and not knowing how to use the programming language correctly to create a program (Winslow, 1996).

DeFino and Bardzell(2006)A **computer program** is a series of instructions written in some computer languagethat performs a particular task. Many times, beginning students concentrate solely on the language code; however, quality software is accomplished only after careful design that identifies the needs, data, and process and anticipated outcomes. For this reason, it is critical that students learn good design techniques before attempting to produce a quality program. Design is guided by an **algorithm**,which is a plan of attacking some problem.

Tony Jenkins analyses many factors that could contribute to the difficulty of programming. These are:

1. Multiple skills

Programming is a complex activity, which involves multiple skills including problem solving, use of IT and a hierarchy of knowledge and abilities. Problem Solving is not trivial and requires component skills including creativity; decision-making; identification of the central issues; recognition of relationships, familiar situations and patterns; development of an algorithm and the translation of the algorithm into executable code.

Students also learn additional IT skills for example editing, compiling, use of debugging tools, saving, loading, combining files and dealing with the IDE interface.

Detailed knowledge of the language syntax is required, and the obscure error messages that compilers often generate are difficult for beginners. Extreme accuracy is necessary. To write programs, students need to analyze problem statements, synthesize solutions and evaluate whether they meet the specification.

2. Conceptual of programming

Constructivists argue that learning involves the creation of meaning by students as they interact with their learning environment, relating new knowledge to existing knowledge and integrating it into their conceptual framework. Programming has little to relate it easily to the familiar; it is largely abstract and thus difficult to relate to existing knowledge.

Programming is a subject that builds continuously. If a student fails to grasp a particular concept, then it can become increasingly difficult to catch up and the pace of the teaching is often driven by the curriculum, not the learning of the student.

2.1 Research methods

Mugenda (1999) explains that a researcher needs to develop instruments with which to collect the necessary information. This study used questionnaire, observation and interviewee as instruments of study. A questionnaire consists of a number of questions printed or typed in a definite order on a form or set of forms (Kothari, 2004). The number of students in one class is 65. Because of limitation of resource in our study, we were select only 21 students in sample. A group of 8 questionnaires was prepared and distributed to 21 selected students. The questionnaire was collected from the students for further analysis.

Observations was takes place to assess the overall students interaction among themselves and their interaction with teachers during programming course delivered in the class. This was done by five observers: three of them were ourselves and the remaining two were selected from among the targets students themselves. Finally, a sample of students who could express their feelings without fear were interviewed to get supplementary data to those already collected through observation and questionnaire.

2.2. Proposed Action/action taken

- Planning tutorial /training
- Preparing manual (work sheet)
- Handout was distributed to the sample students
- Lecture methods was used sometimes to effect an attitudinal changetowards programming
- Preparing questionnaire

✓ Before intervention(pretest)

✓ After intervention(posttest)

- Choose one of the reflective activities
- Review assessment

2.3. Implementation of action/ intervention

2.3.1. Questionaries' before intervention

Q1. What is your basic problem with C++ programing?

Table 1

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Writing code	6	28.6	28.6	28.6
	Understanding problem	15	71.4	71.4	100.0
	Total	21	100.0	100.0	

From the above table1 illustrate those basic problems of students with programing were lack of understanding the problem which accounts 71.4% from sample was taken. Based on the above data the second basic problems was lack of writing code which accounts 28.6% from the sample was taken.

Q2. What caused challenge in C++ during class

Table 2

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	no awareness of programing	8	38.1	38.1	38.1
	consider C++ as difficult	5	23.8	23.8	61.9
	not related to real life	8	38.1	38.1	100.0
	Total	21	100.0	100.0	

From the table2 we recognize that the reason C++ become challenging to students were students background had no awareness of programming that accounts 38.1% and C++ programming is not relate to any real life which accounts 38.1%. This makes C++ challenging to students during their class session.

Q3. How do you see the benefit of the computer programming?

Table 3

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	High	3	14.3	14.3	14.3
	Medium	2	9.5	9.5	23.8
	Low	7	33.3	33.3	57.1
	I don't know	9	42.9	42.9	100.0
	Total	21	100.0	100.0	

Table3 shows that most of students were not consider why they learn C++ in class and use of C++ in their real life. Most of the students said that I don't know (42.9%) and low (33.3%). This indicates thatstudent recognizes the usage of C++ as low and this may because of C++ is considered as difficult.

Q4. How do you want the teaching approach to be for C++ programing?

Table 4

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	long lab hours	5	23.8	23.8	23.8
	long lecture hours	12	57.1	57.1	81.0
	equal lab and lecture	3	14.3	14.3	95.2
	I don't know	1	4.8	4.8	100.0
	Total	21	100.0	100.0	

From the above table4, we realize that teaching approach for C++ was long lectures hours in the class which accounts 57.1% and long lab hours which accounts 23.8%. This data indicates that students were spent most of the time in class not in the lab session.

2.4. Questionaries' After intervention

Q1. What is your basic problem with C++ programing?

Table 5

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Writing code	16	76.2	76.2	76.2
	Understanding problem	5	23.8	23.8	100.0
	Total	21	100.0	100.0	

From Table1 we see that student's response after they got training, the main problems is writing code which accounts 76.2% and the second problem is understanding problem is 23.8%. From this data we conclude that students were understood the problem well and now their basic problem was shifted to writing code.

Q2. What caused challenge in C++ during class

Table 6

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	no awareness of programing	6	28.6	28.6	28.6
	consider C++ as difficult	11	52.4	52.4	81.0
	not related to real life	4	19.0	19.0	100.0
	Total	21	100.0	100.0	

From the above table2 we observe that after training the reason C++ become challenging in the class was considering C++ as difficult which accounts 52.4%. From this data we summarize that we must going to change the attitude of students on C++ programming.

Q3. How do you see the benefit of the computer programming?

Table 7

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	High	11	52.4	52.4	52.4
	medium	7	33.3	33.3	85.7
	low	3	14.3	14.3	100.0
	Total	21	100.0	100.0	

From the above table3 we consider that, after training students were understood the benefit of C++ programming is become high and medium which accounts 52.38% and 33.33% respectively. According to above data after training Students were got where they can be use C++ programming.

Q4. How do you want the teaching approach to be for C++ programing?

Table 8

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	long lab hours	12	57.1	57.1	57.1
	long lecture hours	4	19.0	19.0	76.2
	equal lab and lecture	5	23.8	23.8	100.0
	Total	21	100.0	100.0	

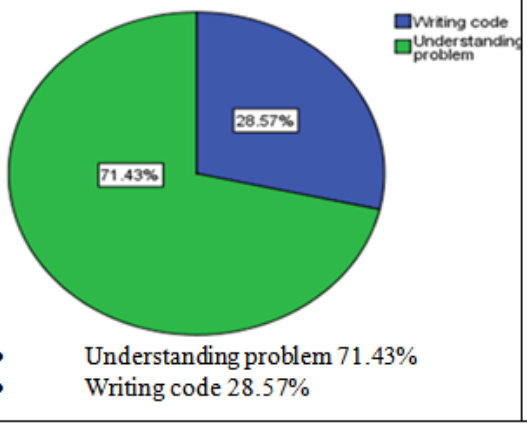
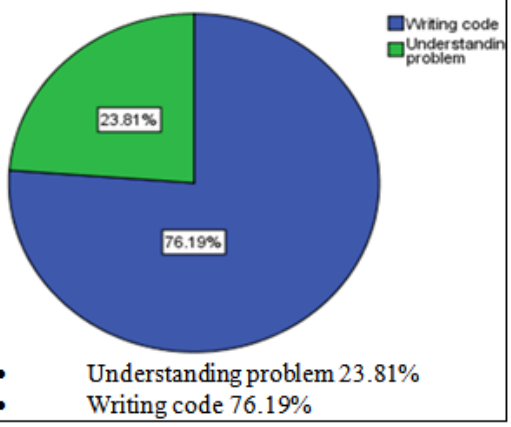
As indicated on the above table8, after the students got training the way C++ programming delivered must long lab session and equal lab and session which accounts 57.14% and 23.81% respectively. This indicates that teaching approach for C++ programming should follows long lab hours favorable for students.

Q5. What do you want to enhance your skill in C++?

Table 9

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Defining problem	3	14.3	14.3	14.3
	Programming skill	8	38.1	38.1	52.4
	writing algorithm skill	10	47.6	47.6	100.0
	Total	21	100.0	100.0	

From the above table9, we recognize that after students trained on C++ programming they want to enhance/ improve writing algorithms skill and programming skill which accounts 47.6% and 38.1% respectively.

No	Pre- intervention	Post intervention
1	 • Understanding problem 71.43% • Writing code 28.57%	 • Understanding problem 23.81% • Writing code 76.19%

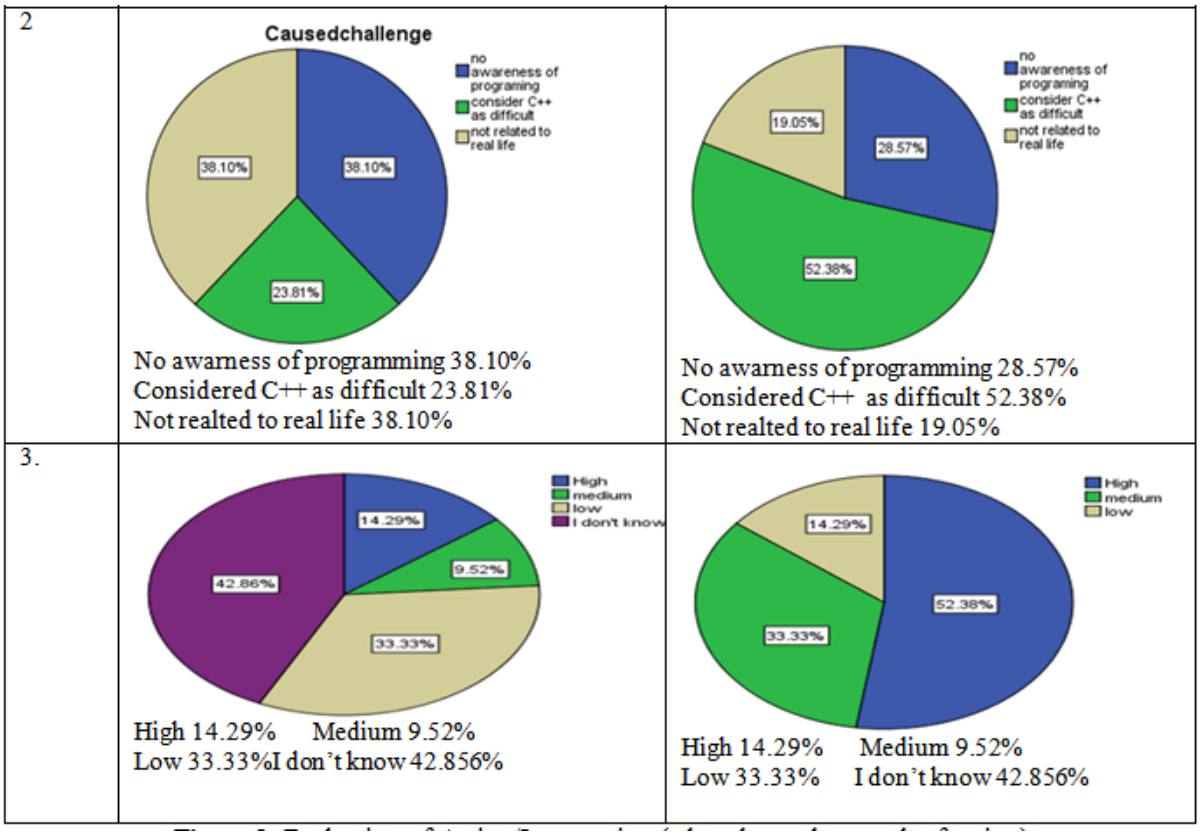


Figure1. Evaluation of Action/Intervention (what changed as result of action)

III. Conclusion

The overall results from this Action research indicates that students post intervention are significantly more effective than pre- intervention. There was definite much difference between the test scores pre-intervention and post-intervention in students. However, results showed that students perceive programming as being difficult and not related to real life in pre-intervention but after post-intervention was takes students enhance their programming skills and move to the next steps as indicated on pre- andpost-intervention test score the difficulty was understanding problem in pre-intervention and this was enhanced as result of action taken and later after intervention the difficulty is changed to understanding algorithms and converting it to coding.

When programming was introduced in later stages, students had a better knowledge on basic concepts in programming and could demonstrate those skills better. This would have resulted in better overall performance. However, the results show a significant improvement when were took tutor of programming later.

In the pre- intervention, most of the programming teachers spent their time on the lecturing students and this makes students programming was difficulty and boring during lecturing in the class. The other point is students are not reflecting their feeling on the lesson and this indicates that participation of the students was very less.After intervention most students need long lab hours than long lecture hours. This is put a room for the students to practice and discuss together.

Involvement in a programming contest effectively motivated student to learn and be interested in programming. With the clear goals of creating and presenting their program, students voluntarily learn programming. If their program does not run, they will seek, through trial and error, to naturally solve the problem using their own skills, which will further develop their programming skills. Students who have completed the program creation and presentation gain confidence and the satisfaction of accomplishment.

Although this Action research has provided some valuable insights, results should not be unconditionally generalized due to the small number of students who participated in this study. It is recommended that the study should be replicated and involves a larger sample of participant's studies over a longer time.

References

[1]. L. Williams and R. Kessler, Pair Programming Illuminated: Addison-Wesley, 2003.
[2]. L. Williams, R.R. Kessler, W. Cunningham, and R. Jeffries, Strengthening the case for pair programming, IEEE Software, 17(4), July – Aug. 2000, pages 19 – 25, 2000
[3]. K. Beck, Extreme Programming Explained: Embrace Change: Addison-Wesley, 1999.
[4]. P. Abrahamsson, J. Warsta, M. T. Siponen, and J. Ronkainen, "New Directions on Agile Methods: A Comparative Analysis," International Conference on Software Engineering, 2003.
[5]. Radermacher, A., Walia, G. "Improving Student Learning Outcomes with Pair Programming" Proceedings of the 8th International Conference on Computing Educational Research - ICER'2012. September 10-12, 2012 Auckland, New Zealand. pp. 87-92
[6]. Blumenstein, M. (2004). Experience in teaching object-oriented concepts to first year students with diverse backgrounds. Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04) [electronic proceedings].
[7]. Buck, D., &Stucki, D. (2001). JkarelRobot: A case study in supporting levels of cognitive development in the computer science curriculum. Proceedings of the SIGSCE Technical Symposium on Computer Science Education, 16-20.
[8]. Dunican, E. (2002). Making the analogy: Alternative delivery techniques for first year programming courses. In J. Kuljis, L. Baldwin & R. Scoble (Eds), Proceedings from the 14th Workshop of the Psychology of Programming Interest Group, Brunel University, June 2002, 89-99.
[9]. Miliszewska, I., Horwood, J., Tan, G., &Venables, A. (2004). Gender bias in computing? – Student perspectives. Proceedings of the Joint International Conference on Informatics and Research on Women in ICT (RWICT), Kuala Lumpur, Malaysia, 1135-1146.
[10]. Jenkins, T. (2002). On the Difficulty of Learning to Program. 3rd Annual Conference of the LTSN Centre for Information & Computer Sciences, Loughborough, LTSN-ICS.
[11]. Bloom, B. S. . . (1965)Taxonomy of Educational Objectives London : Longman.
[12]. Dijkstra, E. W. . . (1989) On the Cruelty of Really Teaching Computing Science. Comm. ACM 32: 1398-1404.

BayisaNateaSima. "AComparative Analysis of Improving Computer Programming Skills of Information Technology First Year Students at Ambo University - An Action Research", IOSR Journal of Computer Engineering (IOSR-JCE) 21.6 (2019): 08-15