# Programming Fundamentals (CS-161)

## with Iterative Project Based Learning

# Introduction to
# Programming and Problem Solving Skills

**With C++**


**Projects:**
**Console Based Business Application**
**Console Based Tiled 2D Game**

Dr. Awais Hassan | Maida Shahid | Laeeq Khan Niazi

**Message From Chairman (Prof. Dr. Muhammad Shoaib)**

Programmers are the people who shape up the future. They create almost everything we use in this digital world.

You must feel proud that you are going to be a part of this journey. No doubt it is a difficult one but remember you are not alone. We are here for you every step of the way. We have highly skilled and dedicated staff here, who would do everything in their power to help you reach a renowned national or international software house or start your own one day.

In order to become a skilled programmer, you will have to practice a lot. You cannot become a programmer overnight. But dedication, motivation and hard work are the key aspects that will definitely get you there. Many times you will be stuck on a problem, maybe even for days. At that time, you will need to pick yourself up and keep moving forward.

Always remember you are smarter than you know, braver than you imagine and more capable than you realize. Every day, try to do a little bit better than you did yesterday. Know that tough times never last, but tough people do. It takes courage to keep trying. Every step you take, every line of code you write, every problem you solve will move you closer to your goal. The more problems you solve, the more confident you will become.

Your biggest challenge in this journey will be to stay focused and diligent about learning and keep practising. If you keep solving the problems given to you with determination then I promise you that you will be among the best Programmers of the world and will make Pakistan proud one day. In Sha Allah!

**Message From Program Mentor (Prof. Dr. Shazia Arshad)**

This course is designed based on student centric policy; it means all the contents, lab work, projects are developed while considering the cognitive and practical challenges that students face. However, we believe a book, an instructor, or a course can only provide you the guidance and a roadmap to follow, with engaging and motivating contents but none of these can make you a programmer until you write a lot of code by thinking through your own mind. This includes writing thousands of lines of code, thinking day and night, solving hundreds of problems, and developing a lot of projects over the course of years.

In this process, you will make mistakes, generate a lot of errors, search for solutions, debug broken code, fix them, do experiments, and find better solutions. Finally you will develop computational thinking skills that allow you to break larger problems into smaller parts and transform your vision into reality.

In short, it is not an easy way, you will pass through a lot of different stages and emotions like being bored, frustrated, overloaded, optimistic, and excited. Sometimes you will literally think about quitting and you will assume others are the masters of programming. Remember, the key is consistency and keep doing the hard work because a day will definitely come when you will feel that now I can solve any type of problem that you will encounter. Also, if you are at a low level of energy or emotions you must immediately consult with your instructors and they will definitely guide you because everyone goes through with what you are experiencing, during his journey to become a successful programmer.

***"Tell me and I forget, teach me and I may remember, involve me and I learn."***

**(Benjamin Franklin)**

**Our Mission: Why we do It**

To be an international partner in computing education, research and development with our graduates impacting the society as computing professionals and entrepreneurs demonstrating professional integrity and leadership.

**Expected Student Outcome: What we Produce**

The objective of teaching first year programming is to produce professionals that can develop softwares to solve real world problems. However, the traditional courses focus too much on the teaching of contents rather than producing the skills. The outcome of such courses fails to generate confident students that can use their knowledge to develop practical applications. The computer science department of UET, in its first year, truly focuses on producing excellent computer programmers with strong fundamental knowledge and practical skills that give them ability to: 1) develop softwares while following industry standard practices and guidelines 2) learn new technology stacks 3) participate in the selection process of top software companies 4) and start freelancing.

**Our Methodology (Shorter Version): How we Do It.**

On the first day, we give a vision to our students of what kind of real world problems they will solve after successful completion of first year. We motivate students to define their own vision and identify their target software applications. We help them to rationalise their goals and after every four weeks, they review their vision based on their learning. Every week starts with a new concept that extends the capabilities of the previously developed application. Further, each lecture starts with a big question (a real world problem), fundamental knowledge to solve the problem, practical presentation of learned concepts, challenges to reinforce knowledge and skills, and finally, a way to the answer of the big question (how to utilise this knowledge into projects) . The journey includes two types of real world software applications: Arcade type Games and Business Applications. The requirements of these projects are drafted carefully so they include almost all types of complexity that is usually encountered during actual software development. According to their vision, students will choose the project of their own choice. After every week of learning they will extend their projects and move closer to their vision.

**The Challenge**

After 20 years of programming and 15 years of teaching programming experience, in our opinion, programming is a highly creative ability that should be constructed gradually with practice rather than taught or conveyed.

Edgar Dale, an American educator, elaborated this with his "Cone of Learning".
He believes that in a two-week's period, we recall only 10% of what we read, 20% of what we hear, 30% of what we see, 50% of what we hear and see, 70% of what we say and write, and 90% of what we really participate in.
Therefore, in the long run, the more students physically and mentally, participate, the more they learn and remember.

For instructors, it is easy to teach concepts, syntax, and the working of any programming construct. However, it is incredibly laborious to engage students and teach them how to think and approach any problem (meta-thinking). Usually, students develop this knowledge through the labor of practice. It is only possible if they are intrinsically motivated rather than the grades forcing them to program. We can reap the full benefits of the competency-based approach with every concept that students learn, every skill they practice, every assignment they perform, and every program they code, if these bring them closer to their final goal. Also, it should be evident to them that they are prospering and progressing after every engagement with a learning component.

However, there is a big challenge to prepare a teaching resource that effectively teaches and produces these competencies because the current curriculums are designed with a knowledge base approach rather than competency based approaches. Usually, to produce a competency in a student requires to take a set of courses. In most cases, it is not possible to teach them in a single subject. Therefore, these high-level competencies demand to be divided into micro competencies, according to the requirement of the subject. In this curriculum, we divided high-level competencies into smaller subject competencies that not only fulfill the needs of the subject but also give a complete skillful and holistic experience to students.

**Teaching Challenges and Our Methodology**

**Teaching Challenges:**

When students kick start their journey in computer programming, they face two types of challenges. First is learning the syntax of the language and second is application of language (problem solving skills). Usually, students easily master the syntax of any programming language but they struggle to apply the syntax to solve real-world problems. Based on our 20-years teaching experience, we believe there are five primary challenges in learning programming:

1. Traditional courses are more focused on syntax rather than on building the problem-solving skills of the students.
2. Lectures focus more on conceptual knowledge and they lack a practical mapping to the taught contents.
3. Students fail to relate the use of classroom contents to actual real-world problems.
4. Students could not appreciate why they are learning any specific concept and how it fits into the larger picture.
5. The learning contents are usually too heavy and less practical that it becomes very difficult to digest

**Our Pedagogy and Teaching Framework:**

Every teacher, for every class, thinks about how to start a new topic, or continue the previous one, how to push them to think, how to ignite the meta thinking, how to engage students, how to evaluate them, how to give feedback to them and many other things. Many of us do it in our head, many others do it unconsciously and sometimes we just ignore it and jump into the class. However, it is a well known fact that if we plan the class consciously and have a strategy to not only teach but also to internalize a concept, it brings better results in terms of student understanding and skills.

We divide contents and teaching activities into following components.

1. **Big and Primary Questions: Why we should learn the concept.**

Motivation is a decisive factor of students' learning which determines how much effort and dedication a student shall put for learning (Brophy, 2013). A typical coursework tries to motivate students through extrinsic motivation (grades) but intrinsic motivation is necessary to do any type of creative and complex learning.

In this course design, to motivate our students intrinsically, every module starts with a Big Question. The question challenges students' show them a problem and ask them how to resolve the problem with their previous knowledge. Then, every lecture starts with a Primary Question, "*know why*", linked with the Big Question. Once they understand what is their final goal, they get conscious and appreciate how a minor learning component takes them towards their destination. This will help them visualize how the smaller pieces of knowledge and skills fit together to solve a big problem. Therefore, every lecture starts with a Problem that challenges the boundaries of their existing knowledge. When they realize the limitations of current knowledge and the need for new concepts, this is the time to deliver the concept. (This is what we call the "Aww" moment for the students).

2. **Contents with Constructivism:**

New Knowledge should always be based on previous knowledge but sometimes we ignore this fact and our contents do not build upon previous knowledge or at least it is not linked consciously with the previous knowledge. It is important to understand what kind of demographic is sitting in our class and what kind of initial base knowledge they have because, whatever we are trying to teach them, it should be based on their existing **mental models**. Therefore, we can not blindly follow the course outlines of caltech/MIT/Oxford. I am not saying we should not benchmark them; we should; but we should not follow those as it is. We have seen that the best content fails if they

are not student centrics. Our main focus is to link the lectures with each other and build the knowledge from the existing knowledge of students. Therefore, while preparing contents we took utmost care to link each concept with the next concept and, therefore, a careful reader may feel the flow of contents is a bit different from legacy course work.

While designing the course work and presentation, we also take care of **cognitive load**. Recent research has shown that the cognitive load is also a major hindrance in learning the contents. Human cognitive architecture has demonstrated the limited nature of working memory of a human in terms of information retention time and capacity. The maximum capacity of items that can be held simultaneously by working memory is four plus or minus one. The greater the items in working memory, the greater will be the cognitive load on a student. When items in working memory are transferred to long term memory, they form a cognitive schema. These cognitive schemas organize the knowledge as well as reduce the load on working memory. A skill is developed when information is stored in cognitive schema and retrieved when it is required.

The excessive cognitive load, course content design, teaching and assessment methods are one of the major factors that restrict students from learning the art of programming **[Sarpon 2013, A. Robins 2003]** .Researches have shown that the current courses on programming fundamentals put high cognitive strain on novices **[Kiesler, 2020]**. One of the major objectives of this course is to reduce the intrinsic cognitive load of a learner by helping them to construct schemas that employ related elements.

In this course, we divide knowledge into chunks; each chunk does not consist of 15 minutes of teaching. After 15 minutes, we present a working example that shows learners how to apply this knowledge on real world problems. After that, to reinforce the concept, the learner will get a chance to apply the learned concept as class activity that helps them to take things from short term memory to long term memory. Also, they get feedback from the instructor on the solution.

**3. Engagement and Active Learning:**

When contents are delivered, students should apply those contents to internalize the knowledge. Therefore, in every lecture, after delivering the required knowledge, we design class engagement activities. These activities allow students to understand the concept, raise the questions, and clarify any misconception. Once the activities are performed by students, the teacher presents the solution of activities that reinforce the knowledge and highlight the mistakes of students.

**4. Hands-on-Experience (Skills):**

Traditionally, universities follow the Knowledge-Based Learning (KBL) approach [Whitehall 1994], which has been used a lot in teaching different subjects. However, with the demanding skills and competencies, KBL failed to deliver according to the expectation [Clear 2020].

There are three primary reasons:

1. Focus on content rather than on students
2. Lack of motivation
3. Less or no focus on developing skills and competencies.

In the KBL approach, the students are bombarded with a large amount of knowledge in the form of lectures and slides and sometimes the knowledge that they do not even require. Students cram contents to pass their exams that help them to secure their degrees but it does not guarantee real world skills [Begel, 2008].

On other hand, it is highly desirable from a computer programmer to practically show their skills and apply their knowledge. Therefore,the primary aim of this course is to shift its focus from KBL to CBL (Competency Based Learning). To achieve this, for every theoretical content, there are corresponding practical examples, targeted projects, defined skills and simple to complex challenges. These artifacts help students to master not only theory but also practical and hands-on skills that will be used to solve real world problems.

We have tried our best to map lab contents with theory knowledge. Each lab consists of close ended problems and open ended problems. In close-ended problems, students follow the predefined steps (some time fully and some time partially) to solve the problems. Once they are comfortable with the syntax and the knowledge, now, the time is to think and solve open ended problems. These problems are designed in the way that students need to think through without any definite given steps. The solution could be different from student to student. However, all these problems do not go beyond the delivered knowledge.

5. **Big Picture (context) : Progression Keeps Student Engage**

The project based learning approach keeps students engaged in class as each concept helps them to add some value into their project and take them near to their vision. Students are strongly involved with the contents and teachers when they experience that they are progressing and moving toward their end goals. To engage students, the lecture contents also include class activities (at least two in a single lecture) that includes: engaging questions, activities and problems.

6. **Feedback:**

*Feedback is a compelling influence on learner achievement. When teachers seek or at least are open to what learners know, what they understand, where they make errors, when they have misconceptions when they are not engaged- then teaching and learning can be synchronized and powerful. Feedback to teachers makes learning visible* [Hattie, J. (2009)]

Therefore, the course is designed to give feedback at four different levels. During lectures, students are given a practice problem that they try to solve with their existing knowledge and most of the problems involve **"How to think"**. Then , the class activity helps them to see what they have done wrong. Usually, the student themself identify the mistakes and correct them. Second level of feedback is given when they submit practical tasks, the feedback automatically given through the unit test cases. Third level feedback given by the peers, GA, and seniors. A mechanism has been designed to relate a group of volunteer students from senior bataches to juniors that help them and give feedback during lab practice. Fourth level of feedback is given through the regular assessment methods like Quizzes, assignment, presentation etc.

**Contributors:**
Special thanks to Muhammad Irzam Liaqat for designing the Lab Manuals and Muhammad Usama Ijaz for recording the video lectures.

| Table of Abbreviations | |
|---|---|
| **WP** | Working Problem |
| **CL** | Close Ended Problems |
| **OP** | Open-Ended Problems |
| **CP** | Complex Problems |
| **CA** | Class Activity |

**Course Outline**

The goal of this course is to develop 2 types of real world software applications: Arcade type Games and Business Applications. In order to achieve this, the contents are divided into 13 modules. These modules can be covered in the duration of one week or more. Each module starts with motivating the students on why they should learn this module and what is its role towards the overall learning goals. Further, each module is divided into small biteable skills (Hands-on Experience). We call these skills biteable because they are easy to absorb and understand. Each skill starts with a problem and raises a question in the students' minds that engages them in the thinking process before delivering the actual contents (Construct Knowledge). The contents of the module include only those contents which are relevant and necessary to master that skill. After that, there is a complete working example to show the students how to approach and solve the problem by applying the required knowledge. This follows a detailed discussion and explanation of each step of the solution. Then at the end of each skill, a list of challenging problems related to the skill is given that help students to achieve mastery level. The list of Modules are:

1. *Motivation to Learn Programming and Computer Science: First Program*
2. *Converting Input into Output With Expressions*
3. *A Gentle Introduction to Reusability - Repetition and Conditions*
4. *Code Reusability with your own as well as Functions of other developers .*
5. *Decision Making with Complex Conditional Statements.*
6. *Repetition and Loops to solve complex problems.*
7. *Project and Feedback Time: Design, Develop, Document and Present the First Software Project.*
8. *Array: Time to Handle Large Number of Records.*
9. *File Handling Time to Store Records Permanently.*
10. *Reading a very large file: Role of Dynamic Memory Allocation, Deallocation and Pointers*
11. *2D Arrays and Game Project.*
12. *Complex Problem Solving and Project ShowCase Week*
13. *Python Time - Experiencing any other language*

After covering each module, students will be able to develop a small part of a console-based business application and 2D game. Next module will help the students to appreciate the contents when they will come to know how this helps them to further improve the projects. After completing 11 modules, students will be able to develop full fledged console-based business applications and 2D console-based games. During this course of study, students will develop problem-solving skills along the way.

Because after 20 years of programming and 15 years of teaching programming experience, we have come to this conclusion:

***"Programming can only be learned by experimenting and executing your own code, not by just reading someone else's."***

**Module 01: Motivation to Learn Programming and Computer Science: First Program**

We believe students can learn better if they have a clear vision and objective in front of them. They should know beforehand what kind of capabilities they are going to earn after successful completion of the course. This week motivates students to learn and put all the hard work coming ahead. Also, they will get a gentle introduction to programming, its magic, GUI, CLI, and setting up the environment by running the first computer program. They will understand the role of hardware, computer languages, and programs; How do the programs work and what is the basic property of a program. Students will be able to print a message on the screen by developing and executing a computer program. This week enable the students to output a header for their business application and maze/grid for their future projects (vision)

**Big Question:** Why should I learn Computer Programming, How to Start it and What kind of skills I will have after the first year and How to Instruct Computers ?

| Primary Questions (Why of Module) | Contents (Construct Knowledge) | Engagement (Active Learning) | Hands-on Experience (Skills) | Big Picture (Context) |
|---|---|---|---|---|
| What is the Computer and Its Role in Our Daily Life?<br>Why Study Computer Science?<br>How to Save and View Our Work in Computers.<br>How Computer Works.<br>Setting up the Vision.<br>How to Instruct a Computer to print something on the console? | **Explain** Application of Computing.<br>**Explain** Computer Hardware and Its working: IPO Cycle.<br>**Explain** Computer Softwares and Its Working<br>**Explain** Communicating with Computers: CLI and GUI.<br>**Explain** Computer, Languages, Compiler and Interpreter. | Writing and Saving first program. (WP)<br>Compiling and Executing first program. (WP)<br>Printing menu for a business application. (WP)<br>Print a Line of Asterisks. (CL)<br>Print Geometric Shapes. (CL)<br>Printing a Big Alphabet. (CL)<br>Printing a Game Character. (CL)<br>Printing a Game Maze. (CL) | Creating, Storing, Location files/directory through windows explorer and CLI<br><br>Write, compile and Execute a program to print the output on the screen.<br><br>Use special directives to control output on the screen. | First screen of the Game Project.<br><br>Print PacMan Maze / Spaceship / Flappy Bird / Mario World.<br><br>Main Menu of the Business Application. |

List of Problems: Practices and Feedback

**Practice Problems:** A Big Name with Alphabets; Game Stage; Business Application Home Screen.
**Feedback Mechanism:** Instructors and Teacher Assistants will be available in the Lab for real-time instant feedback to the students queries.

## Projects: Screenshots

### Business Application

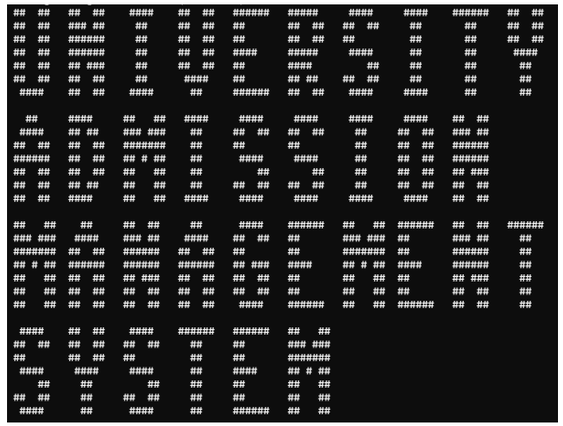

Figure 1: Project Header

### 2D Game

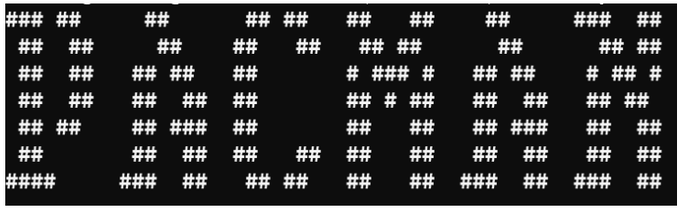

Figure 1: Game Header

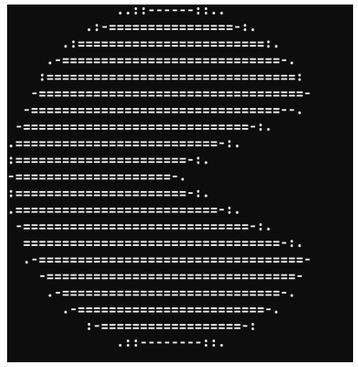

Figure 2: Game Character

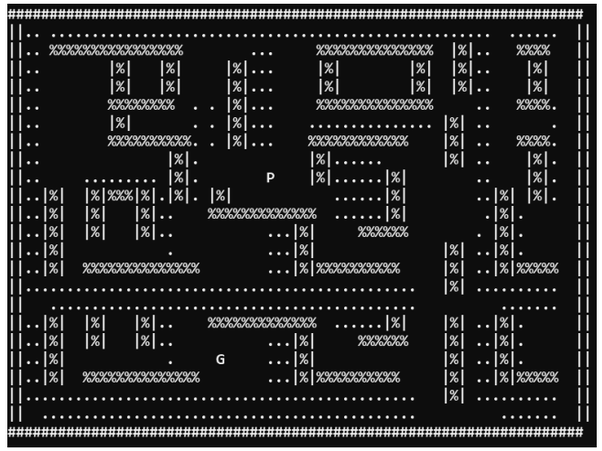

Figure 3: Game Maze

**Module 02: Converting Input into Output With Expressions**

It is very common for programmers to write programs that take input, store it into the memory, convert it into the desired form, and then show it on the computer screen. For example, take the velocity and distance as input, calculate the time, and print it on the screen as output. Students should be able to understand the role of memory, how data is assigned and stored in the memory, and write C++ programs that take input from the user, apply mathematical expressions (precedence rule), and give output on the Console.

**Big Question:** How to develop computer applications that can convert one form of data to another form of data ?

| Primary Questions (Why of Module) | Contents (Construct Knowledge) | Engagement (Active Learning) | Hands-on Experience (Skills) | Big Picture (Context) |
|---|---|---|---|---|
| **Primary Questions**<br>What type of application exists around us that convert input into output ?<br>Why do we need memory ?<br>How to use memory ?<br>How to convert one form of data into another form ? | Concept of Memory<br>Role of Memory<br>Variables and DataTypes<br>Naming Conventions<br>Taking Input from User<br>Assignment Operator.<br>Mathematical Expressions<br>Precedence Rules | Currency Converter (WP)<br>Weight Converter (WP)<br>Area Calculation (CA)<br>MegaBytes to Bits Converter (CL)<br>Calculate Current in a wire (CL)<br>Aggregate Calculator (CL) | Identify the Variables for any given problem.<br><br>Identify the Datatypes of the Variables.<br><br>Write a complete program that converts input into the required output. | UAMS Merit Calculator.<br><br>Make the PacMan/Space Invader/Mario Game object with ASCII Characters and Colours (CP) |

List of Problems: Practices and Feedback

**Practice Problems:** Joyland Ticket Cost; Flower Shop; Lose Weight;Grow Vegetables; Local theatre Charity; Half pyramid; Physics Numericals

**Feedback Mechanism:** Instructors and Teacher Assistants will be available in the Lab for real-time instant feedback to the students queries.

## Projects: Screenshots

### Business Application

```
Enter Student Name...Ahmed
Enter Obtained Marks in Matric..1060
Enter Obtained Marks in First Year..490
Enter Ecat Marks..313
Your Merit Score is..88.804
```

Figure 1: Aggregate Calculation

### 2D Game

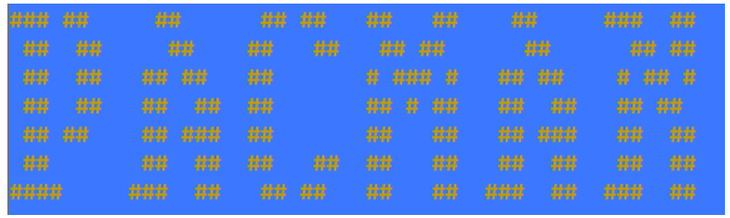

Figure 1: Colored Header

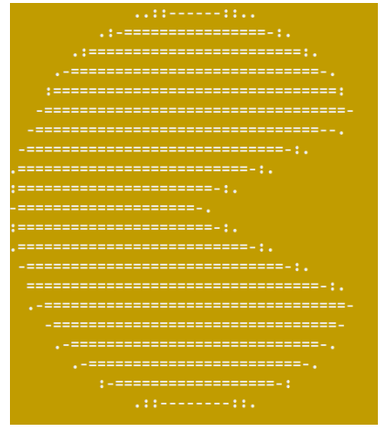

Figure 2: Colored Character

**Module 03: A Gentle Introduction to Reusability - Repetition and Conditions**
In this module students will understand why they need to reuse code and they will be able to write their own function and use the predefined functions. A gentle introduction of IF statements and Loop will be given so they can connect different things and put them into a form of a working application. This will give them a sense of confidence and allow them to understand how practical applications are actually based on the fundamental construct of computer programming.

**Big Question:** How to develop computer applications with reusable code that continuously execute and perform operations based on conditions.

| Primary Questions (Why of Module) | Contents (Construct Knowledge) | Engagement (Active Learning) | Hands-on Experience (Skills) | Big Picture (Context) |
|---|---|---|---|---|
| **Primary Questions**<br>Why do we need reusable code ?<br>Why do we need to write code with conditions ?<br>Why do we have to make applications run continuously ?<br>Why is repetition required. ? | Functions with reusable code.<br>Built in Functions: GoToXy; GetCh.<br>IF Statement<br>IF Block (with curly bracket) and Without IF Block.<br>Comparison Operator<br>Boolean Expressions<br>While Loop | Reuse Code of Alphabets for Printing text at any part of the screen. (WP)<br>Move the name by calling a trail of functions multiple times without loops. (CL)<br>Pass or fail. (WP)<br>Greater number from two. (CA)<br>Even or odd. (CL)<br>Keep printing the name. (CA)<br>Print name for 10 times. (CA)<br>Print menu and exit if user press X. (CA)<br>Move the name from left to right. (CA)<br>Move name with the Arrow Keys. (CL) | Write reusable code using functions.<br><br>Write simple conditional statement that involve single boolean expression<br><br>Write a while loop to run the program continuously and terminate it on some condition. | Move the object of the game, calculate the score, and die on the collision with the enemy.<br><br>Develop a university management system with a menu that allows to add information of two students, calculate and show merit, and give admission to one student with highest marks. |

List of Problems: Practices and Feedback

**Practice Problems:** Build a flower shop with products, discounts and allow the user to calculate his estimated price if he buys X number of Y type flowers on Day Z. Build a Calculator App with functions, loops and conditions. Ask from the user how he want to move a object and do it accordingly: Possible movement options 1) Move Vertically from given location 2) Move Horizontal from given location 3) Move through the KeyBoard 4) Round the Screen. 5) Ask Boundaries to Move an Object
**Feedback Mechanism:** Instructors and Teacher Assistants will be available in the Lab for real-time instant feedback to the students queries.

# Projects: Screenshots

## Business Application

```
************************************************************
*        University Admission Management System            *
************************************************************


Select one of the following options number...
1. Add New Student Record
2. View all Records
3. Show Records in Descending Order
4. Exit
Your Option..
```

Figure 1: Main Menu

```
************************************************************
*        University Admission Management System            *
************************************************************


Select one of the following options number...
1. Add New Student Record
2. View all Records
3. Show Records in Descending Order
4. Exit
Your Option..1
Enter Student Name...Ali
Enter Obtained Marks in Matric..1000
Enter Obtained Marks in First Year..480
Enter Ecat Marks..300
```

Figure 2: If the user pressed 1

## 2D Game

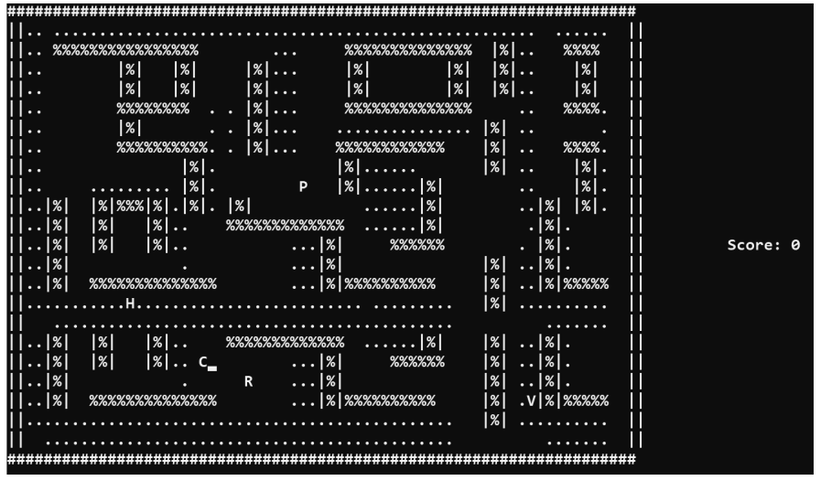


Figure 1: Pacman at the Starting position

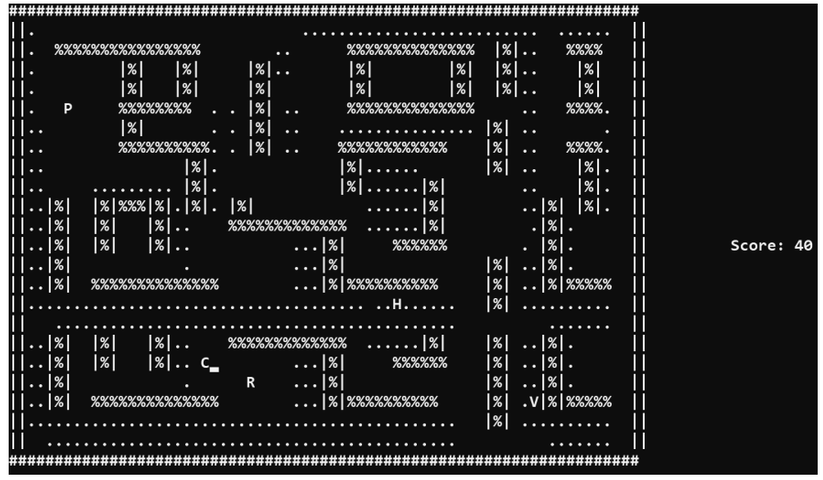


Figure 2: Pacman moved and Collected pills and score increased

**Module 04: Code Reusability with your own as well as other's Functions.**
In this module, students will dig deeper into the Functions. They will be able to define their own functions with zero, one or more parameters as well as they will be able to return the result from the functions. Students will be able to write their code effectively by using pre-defined or user-defined functions.

**Big Question:** How to develop the software applications effectively with reusable blocks of code?

| Primary Questions (Why of Module) | Contents (Construct Knowledge) | Engagement (Active Learning) | Hands-on Experience (Skills) | Big Picture (Context) |
|---|---|---|---|---|
| How to Divide problems into reusable functions ?<br><br>How to pass different parameters to the functions ?<br><br>How to return the result from the functions? | Pre-defined functions.<br>User-defined functions.<br>Return Type of Functions<br>Local and Global Variables.<br>Function Call Stack | Addition of Numbers (WP)<br>Digit Distance (CA)<br>Temperature Converter (CA)<br>Comparison of Integers (CA)<br>AntigenChecker (WP) | Solve problems by using User-defined Functions and Pre-defined Functions.<br><br>Differentiate between Void and Value Returning Functions.<br><br>Distinguish between Local and Global Variables | Categorize the movement, collision and score calculation functionalities of the game into user-defined functions.<br><br>Categorize different functionalities and menus of the Business Application into reusable code. |

List of Problems: Practices and Feedback

**Practice Problems:** Find Ancestors and Offsprings; Volume of a pyramid; Leap Year; Leap years between ten years; Calculate Human years, Dog years and Cat years.
**Feedback Mechanism:** Instructors and Teacher Assistants will be available in the Lab for real-time instant feedback to the students queries.
**Quiz:** Quiz 1 will be taken and its solution will be discussed.

# Projects: Screenshots

## Business Application

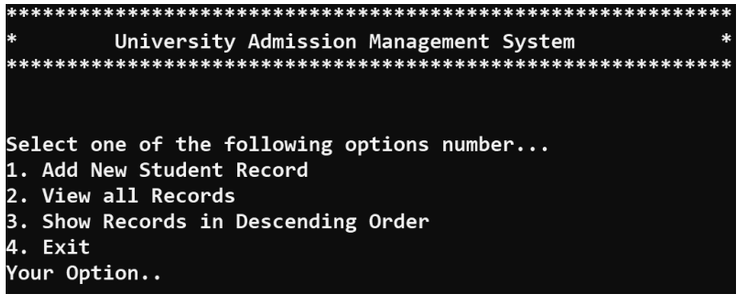


Figure 1: Main Menu

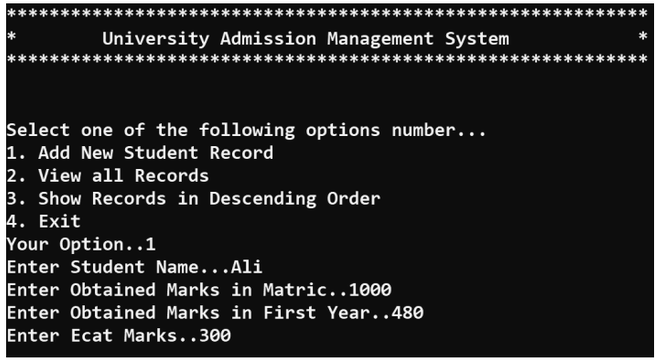


Figure 2: If the user pressed 1

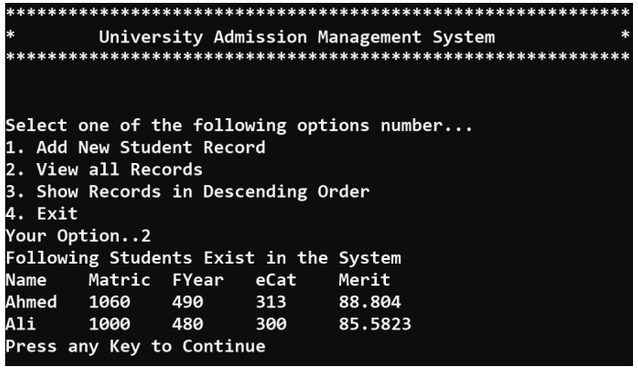


Figure 3: If the user pressed 2

## 2D Game

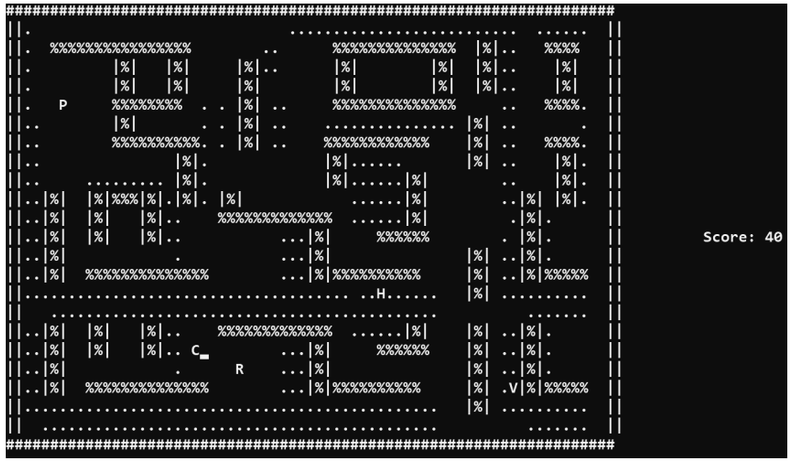


Figure 1: Pacman moved and Collected pills and score increased

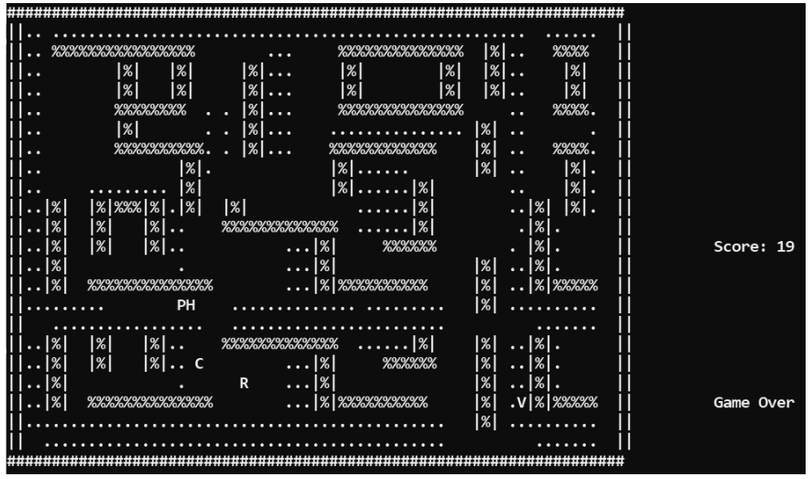


Figure 2: Game Over on Collision

**Module 05: Decision Making with Complex Conditional Statements.**
**Daily life is full of decision making and computer programs solve the problems related to daily life; therefore, decision-making is an essential component of any fundamental programming course. We have already seen decision-making commands that allow us to perform actions based on simple conditions. At this stage, students need to develop skills to write more complex decision-making commands: sometimes actions are performed when multiple conditions are satisfied; any one of the conditions is satisfied out of many; when a condition is not satisfied, even when a combination of all these conditions are satisfied or not. All these types of conditions come under this week.**

**Big Question:** As single conditions are not sufficient for real time applications, how to add complex conditions into software applications ?

| Primary Questions (Why of Module) | Contents (Construct Knowledge) | Engagement (Active Learning) | Hands-on Experience (Skills) | Big Picture (Context) |
|---|---|---|---|---|
| How to Handle Alternative Conditions<br>How to Perform some tasks when Multiple Conditions, any any condition, or combination of different condition are satisfied at same time<br>How to perform some tasks when some condition is not satisfied. | **Multiple Conditions.**<br>IF Else Block, Multiple IF<br>Multiple IF vs Else<br>**Nested If Statements**<br>When Multiple Conditions.<br>Nested-Conditions.<br>Nested IF Statements<br>**Logical Operators**<br>Combining Multiple Conditions<br>Logical Operators.<br>Order of Precedence. | Brilliant Student (WP)<br>Discount on Shopping (CA)<br>Bonus Score (WP)<br><br>Find Largest Number from given Three Numbers (WP)<br><br>Number is Within Range (CA)<br>Raise the Salary (WP)<br>Grading System (CA) | Write complex conditions with logical and comparison operators.<br><br>Draw decision trees from complex real world problems.<br><br>Convert real world problems with complex conditions (decision trees) into the code. | Students will be able to add multiple roles and their corresponding features in Business Applications.<br><br>Students will be able to add collisions between 2 objects, add score on collecting the bonus and remove objects when colliding with other objects. |

**List of Problems: Practices and Feedback**

**Practice Problems:** Perimeters of Selected Shape-Even Odd; Greater Number; AM Going UNI; Valid Name; Medical Store; Fast Food; vowel/consonant/number; Airline Discount; Fine Speed Limit; Temperature Difference; Flower Shop; Sleeping Cat; Speed Info; Area of Selected Shape; Number to Text.
**Feedback Mechanism:** Instructors and Teacher Assistants will be available in the Lab for real-time instant feedback to the students queries.

# Projects: Screenshots

## Business Application

```
**********************************************************
*        University Admission Management System          *
**********************************************************


Main Menu > Login
--------------------
Enter your User Name: Ali
Enter your Password: 123
```

Figure 1: Login Menu

```
**********************************************************
*        University Admission Management System          *
**********************************************************


Main Menu >
--------------------
Select one of the following options number...
1. Add New Student Record
2. View all Records
3. Print the Record in Merit Order
4. Generate Merit List
5. Seat Management
6. Exit
Your Option..
```

Figure 2: Main Menu

```
**********************************************************
*        University Admission Management System          *
**********************************************************


Main Menu > Add Student
--------------------
Enter Student Name.................Khalid
Enter Obtained Marks in Matric.....990
Enter Obtained Marks in First Year..480
Enter Ecat Marks...................250

***** Enter the Name from Following Programs *****
CS      SE      IT
**********************************************
Enter First Preference.............CS
Enter Second Preference............SE
Enter Third Preference.............IT
```

Figure 3: Add Student

```
**********************************************************
*        University Admission Management System          *
**********************************************************


Main Menu > Show All Students
--------------------
Name    Matric  FYear   eCat    Merit   1st     2nd     3rd
Bilal   1020    450     300     83.6039 CS      SE      IT
Umer    1000    430     290     80.7413 CS      IT      SE

Press any Key to Continue..
```

Figure 4: Show all students

```
**********************************************************
*        University Admission Management System          *
**********************************************************


Main Menu > Student Merit List
--------------------
Bilal    Got Admission in CS With Merit 83.6039
Umer     Got Admission in CS With Merit 80.7413

Press any Key to Continue..
```

Figure 5: Student Merit List

```
**********************************************************
*        University Admission Management System          *
**********************************************************


Main Menu >  Seat Management
--------------------
1. View All Seats
2. Increase Program Capacity
3. Add New Seats
4. Previous Menu

Enter the Option Number...
```

Figure 6: Seat Management

```
**********************************************************
*        University Admission Management System          *
**********************************************************


Main Menu >  Seat Management > View All Seats
--------------------
1.      CS      2
2.      SE      1
3.      IT      1

Press any Key to Continue..
```

Figure 7: View All Seats

```
**********************************************************
*        University Admission Management System          *
**********************************************************


Main Menu >  Seat Management > Increase Program Capacity
--------------------
Enter the Program Name  :IT
Enter Number of Seats   :50
Seat Capacity been updated

Press any Key to Continue..
```

Figure 8: Increase Seats Capacity

**Module 06: Repetition and Loops to solve complex problems.**
**What if we have to print some text on screen 100 times? Should we write one hundred commands to display output? All Programming languages offer a faster and more intuitive way to repeat certain blocks of code again and again. These types of language constructs are called loops. Also, in nature there are many complex problems that can be solved by repeating small subproblems; therefore, the skill to write a computer program that repeats a certain set of commands to solve a complex problem is very essential to a computer programmer.**

**Big Question:** How do different software applications continuously execute some tasks like moving a tank in game or fire? Similarly finding all sundays between any given dates

| Primary Questions (Why of Module) | Contents (Construct Knowledge) | Engagement (Active Learning) | Hands-on Experience (Skills) | Big Picture (Context) |
|---|---|---|---|---|
| How to repeat a code for some known number of times or based on some conditions.<br>How to divide complex problems into smaller repeatable sub-problems.<br>How to stop a loop before its time of completion and How to skip a step within the loop.<br>Why do we need nested loops and how to use them? | Counter vs Conditional Loop<br>**Dependent SubProblems**<br>Divide a Problem in Computation Steps<br>Identifying the Pattern<br>Writing a Sub Problem that can be repeatedly solved.<br>**Thinking in Loops.**<br>Best tips for Solving Dependent Sub Problems<br>Nested Loops, Break, Continue | Print a message five times (WP)<br>Take input until -1 (WP)<br>Sum of First 100 numbers. (CA)<br>Factorial (CA)<br>Printing Days for Different Weeks (CL)<br>Printing Shapes with Nested Loops (WP) | Distinguish the requirement between the use of counter and conditional loops.<br>Divide complex problems into smaller easily solvable sub-problems.<br>Differentiate between the continue and break statement.<br>Apply nested loops to solve the related problems | Keep on taking the input from the user if the input is not in correct format.<br><br>Move the Ghost and PacMan using AsyncKeyState function<br>Move the Bullets - enemies. |

**List of Problems: Practices and Feedback**

**Practice Problems:** Digits in a number, Print Table, Print Pattern, Generate Sequence, Triangular Dots, Doctor Appointment, Numbers Percentage, Cargo in each vehicle
**Feedback Mechanism:** Instructors and Teacher Assistants will be available in the Lab for real-time instant feedback to the students queries.
**Quiz:** Quiz 2 will be taken and its solution will be discussed.

# Projects: Screenshots

## Business Application

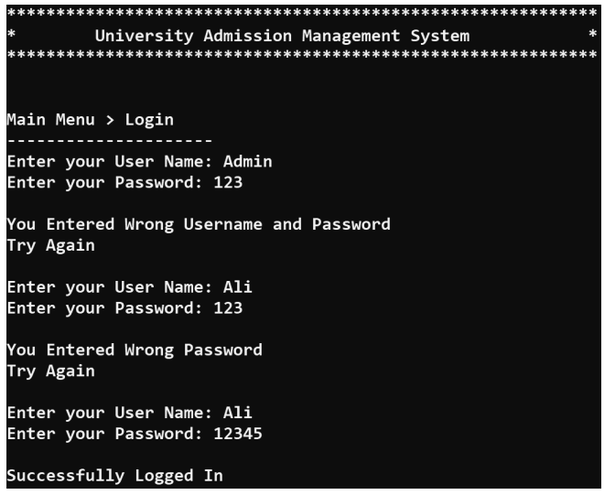


Figure 1: Login Menu (Keep taking the input until it is correct)

## 2D Game

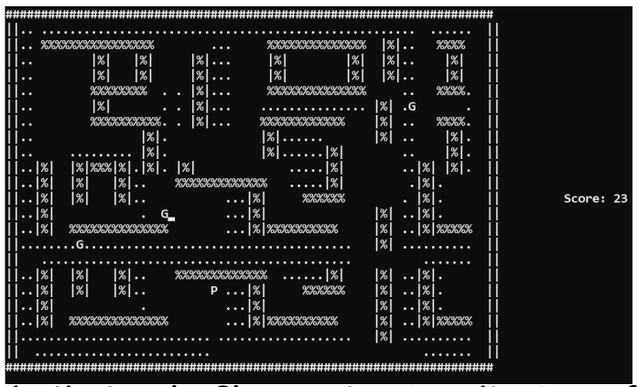


Figure 1: Moving the Player using AsyncKeyState function

**Module 07: Project and Feedback Time: Design, Develop, Document and Present the First Software Project.**
**Time to combine the students' knowledge into working applications: Game and Business Application. Here, students will first divide the project effectively into the functions (design), write the code (develop), make a document to communicate with peers about the project details and finally students will get a chance to show and present their projects to session fellows and their instructors. Students will first do a self-assessment according to given parameters (see Rubric EVS-BL-R-01 and EVS-GM-R-01) after which they will also get feedback from the instructors and be given a chance to improve the feedback in Release 02.**

**Big Question:** How to develop an in-memory business application with a limited number of records ? How to develop 2D maze based simple games ?

| Primary Questions (Why of Module) | Contents (Construct Knowledge) | Engagement (Active Learning) | Hands-on Experience (Skills) | Big Picture (Context) |
|---|---|---|---|---|
| How to Divide projects into functions ?<br><br>How to combine all programming constructs into the game ?<br><br>How to develop a software to solve a Business Problem ? | Software Design.<br>WBS: Work Breakdown Structure and Modularity.<br><br>Self Assessment with Rubric | Development of Grid Based 2D Game: Pacman (CA)<br><br>Development of the Business Application: UAMS. (CA) | Divide the project into the functions.<br><br>Write Software User Manual<br><br>Present the work. | 2D Grid based games. The movement of the player can be controlled through the keyboard. Also, player will get reward/punish on collection coins<br><br>Business application that allow to add 5 records, search, update record, delete record, show record with some processing on its values; for example show the merit of each student |

List of Problems: Practices and Feedback

**Feedback Mechanism:** Students will present their first iteration of both Game based projects and Business Application and feedback will be given to further improve their projects.
**Mids:** Mids exam will be taken and its solution will be discussed afterwards.

**Module 08: Arrays: Time to Handle Large Number of Records.**
**With existing knowledge, if we need to store records of 100 student names then we have to declare 100 variables. The management and processing of these variables are not very easy. Languages provide an alternative way (Arrays) to declare, initialise and process a large number of same-type variables. After getting skills on the Arrays, students will be able to extend their business application of projects to any number (limited to available physical memory) of records. Also, game objects can be maintained more easily with arrays, and the number of enemies, bullets and other moving objects will be increased with the help of arrays.**

**Big Question:** How to extend the capabilities of Software Applications to store an unlimited number of records/objects ?

| Primary Questions (Why of Module) | Contents (Construct Knowledge) | Engagement (Active Learning) | Hands-on Experience (Skills) | Big Picture Context |
|---|---|---|---|---|
| Why scalar variables are not sufficient.<br>How memory stores multiple variables while we can access those with the same name.<br><br>How to store related records of different data types using the concept of the arrays. | **Need and Use of Arrays**<br>When Scalar Variables fails<br>Declaration and Memory<br>Store Values in Array.<br>**String as Arrays**<br>String as Character Array<br>Accessing a Character<br>Processing Character Array.<br>**Parallel Arrays.**<br>How to store different data type information that is semantically the same.<br>Processing Parallel Arrays. | Processing Arrays with Loops. (WP)<br>Find number is in the Array. (WP)<br>Cinema Movie Discount (CA)<br>Even or Odd Length of String (CA)<br>Valid email (CL)<br>Storing username and password in parallel arrays (CA) | Elaborate why scalar variables are not sufficient.<br>Visualise the memory representation of the arrays.<br>Declare and initialize arrays of different data types.<br>Use parallel arrays for storing the related information of different data types. | Users should be able to store any number of records (first the upper limit was at three now it is unlimited). Add at least three type of validation on the different fields (For example, password length; password should contain at least one special character, one number, one capital letter, and one small letter), email should contain @ symbol and dot etc<br><br>Store the maze of the game in parallel arrays. |

**List of Problems: Practices and Feedback**

**Practice Problems:** Smallest from the Data, Count Vowels, Alphabets shuffling, String Length(Even/Odd), Identical Array or Not, Even/Odd Transformation, Common Characters, Colour a Striped Pattern,ATM Machine Pin Number
**Feedback Mechanism:** Instructors and Teacher Assistants will be available in the Lab for real-time instant feedback to the students queries.

# Projects: Screenshots

## Business Application

```
**********************************************************
*        University Admission Management System          *
**********************************************************


Main Menu >
--------------------
Select one of the following options number...
1. Add New Student Record
2. View all Records
3. Print the Record in Merit Order
4. Generate Merit List
5. Seat Management
6. Exit
Your Option..
```

Figure 1: Main Menu

```
**********************************************************
*        University Admission Management System          *
**********************************************************


Main Menu > Show All Students
--------------------
Name    Matric  FYear   eCat    Merit   1st     2nd     3rd
Bilal   1020    450     300     83.6039 CS      SE      IT
Umer    1000    430     290     80.7413 CS      IT      SE
Hassan  980     430     280     79.5152 CS      IT      SE
Zubair  960     410     260     75.9026 CS      IT      SE
Akbar   950     400     250     74.0963 SE      IT      CS
Jabar   940     390     290     76.04   SE      IT      CS

Press any Key to Continue..
```

Figure 2: Unlimited Students

```
**********************************************************
*        University Admission Management System          *
**********************************************************


Main Menu > Student with Descending Order
--------------------
Name    Matric  FYear   eCat    Merit   1st     2nd     3rd
Bilal   1020    450     300     83.6039 CS      SE      IT
Umer    1000    430     290     80.7413 CS      IT      SE
Hassan  980     430     280     79.5152 CS      IT      SE
Jabar   940     390     290     76.04   SE      IT      CS
Zubair  960     410     260     75.9026 CS      IT      SE
Akbar   950     400     250     74.0963 SE      IT      CS

Press any Key to Continue..
```

Figure 3: Students in Descending Order

```
**********************************************************
*        University Admission Management System          *
**********************************************************


Main Menu > Student Merit List
--------------------
Bilal    Got Admission in CS With Merit 83.6039
Umer     Got Admission in CS With Merit 80.7413
Hassan   Got Admission in IT With Merit 79.5152
Jabar    Got Admission in SE With Merit 76.04
Zubair   Did not get Admission  With Merit 75.9026
Akbar    Did not get Admission  With Merit 74.0963

Press any Key to Continue..
```

Figure 4: Students Merit List

## 2D Game

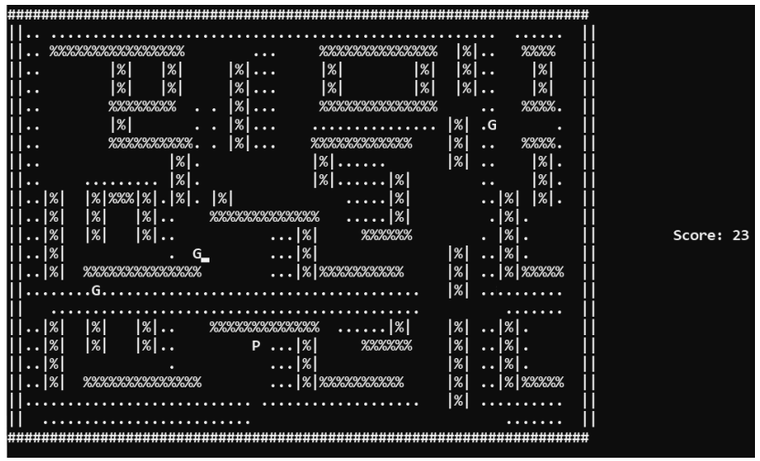


Figure 1: Game Grid stored in Parallel Arrays

**Module 09: File Handling Time to Store Records Permanently.**
**Until this week, students will reasonably mature the business applications and games. However, there is still a drawback and that is if the program is restarted all data goes away. Therefore, we need to store data in secondary storage (hard drive through files) so it remains to persist even when the application has to be restarted. After completion of this week, students will convert their business applications to store data inside the files and get back the data into the main memory (Arrays) so their application keeps performing in the same way as it was previously. Also, they will be able to store the game state at any point and reload it later on when it is required.**

**Big Question:** How to make the project to store data permanently ?

| Primary Questions (Why of Module) | Contents (Construct Knowledge) | Engagement (Active Learning) | Hands-on Experience (Skills) | Big Picture Context |
|---|---|---|---|---|
| How to Store and Retrieve Data from Harddisks to RAM.<br><br>How to store related data in columns and rows:Reading and Writing CSV file structure.<br><br>How to convert projects from temporary storage to permanent storage with minimum changes. | **File Stream**<br>What is the Need of File<br>Five Steps of Writing and Reading<br>Data from File to Array, Data from Array to File<br>**Comma Separated File**<br>Why a Format is Required.<br>Writing information as CSV<br>Reading back from CSV.<br>**Converting Projects through Files**<br>How to update the existing projects using files.<br>Load and Save Data Function. | Writing and Reading Names into the File (WP)<br><br>Sign-up/Sign-in Application (WP)<br><br>Conversion of UAMS with the file. (CA) | Elaborate the need of file handling.<br>Develop a program to read data from the file into the arrays.<br>Develop a program to write data from the arrays into the file. | Students will make their projects to store information into files and then read the information from files. Now, the data will persist when the program is restarted.<br><br>Grid Should be Loaded from the file.<br>Students will add a feature so the state of the game is saved into the file and can be loaded back. |

**List of Problems: Practices and Feedback**

**Practice Problems:** Student Information; Words less than 4 digits; Text Editing Software; Frequency of digits; Birthday Cake; Missing digits from the alphabets; Secret Services; Pizza Delivery System
**Feedback Mechanism:** Instructors and Teacher Assistants will be available in the Lab for real-time instant feedback to the students queries.

# Projects: Screenshots

## Business Application

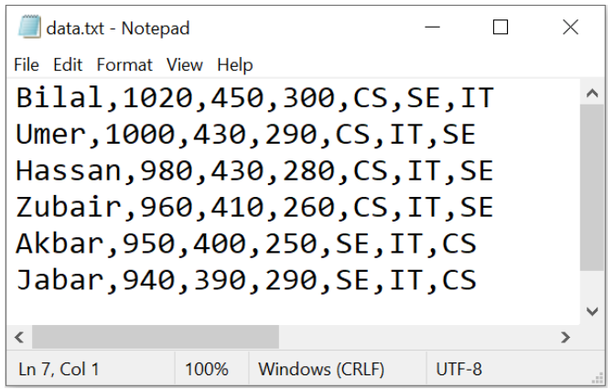


Figure 1: Student information stored in Files using comma separated format


Figure 2: Loaded the Data from the File

## 2D Game

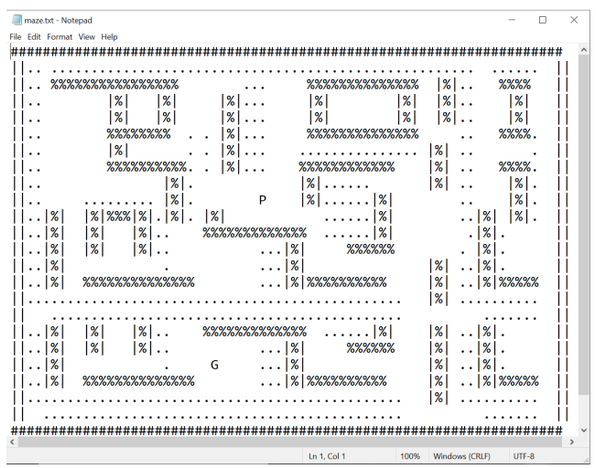


Figure 1: Game maze stored in File


Figure 2: Menu for loading information from file

**Module 10: Reading a very large file: Role of Dynamic Memory Allocation, Deallocation and Pointers**
**We have access to the memory through variables and arrays. These are very useful constructs but sometimes we need to read data from the file that is larger than available primary memory. In these cases, we need to allocate/deallocate memory manually and here the pointers will come into play.**

**Big Question:** Why and when do we need to allocate and deallocate memory manually?

| Primary Questions (Why of Module) | Contents (Construct Knowledge) | Engagement (Active Learning) | Hands-on Experience (Skills) | Big Picture Context |
|---|---|---|---|---|
| How to access memory through memory addresses<br><br>How to Allocate and DeAllocate memory directly<br><br>How to read a large amount of data from a very big file to memory. | **Why to access memory by address**<br>Pointer Variable and its value.<br>Dereferencing the Pointer.<br>Learn to Use Memory Tables<br>Pass by Value; Pass by Reference.<br>Returning Multiple Values.<br>Scope of the Pointer Variables.<br>Stack and Heap Memory<br>Passing and Returning Arrays<br>**Dynamic Memory Allocation**<br>Dynamic Memory DeAllocation<br>Buffer Concept<br>Reading a large file<br>**Project Rubric and Poster Detail** | What is the Output (WP + CA)<br><br>Find a value from a Big Data File (WP) | Explain the access of memory addresses using Pointers<br><br>Differentiate between passing the parameters by value and by reference.<br><br>Write a program that can allocate/deallocate memory dynamically to perform a useful task. | Students will finalize their Business Applications. |

**List of Problems: Practices and Feedback**

**Project Week:** Finalise the individual projects no other lab tasks
**Quiz:** Quiz 3 will be taken and its solution will be discussed.

**Module 11: 2D Arrays and Game Project.**
**Some types of data are easy to process if it is stored in the form of a matrix. For example, if we look at the 2D game grid, it is naturally the best candidate to store into the matrix form (2D Array). This week, students will take advantage of the 2D arrays, store their games into the matrix form, and add more dynamic features. This week, students need to add new functionality into their game projects (see minimum requirement given in EVS-GM-R-01) and do a Self Assessment check.**

**Big Question:** How to store game board, players, enemies and other information more effectively ?

| Primary Questions (Why of Module) | Contents (Construct Knowledge) | Engagement (Active Learning) | Hands-on Experience (Skills) | Big Picture Context |
|---|---|---|---|---|
| When Parallel Arrays are Not Enough<br><br>How to Declare,Access and Process 2D Arrays<br><br>How to Store Grid based games to 2D Arrays | **2D Array**<br>When Parallel Arrays are Not Enough<br>Declaring and Accessing 2D Arrays<br>Processing 2D Arrays<br>**GridBased Game with 2D Array.**<br>Why a 2D Array to Hold Grid Info.<br>Movements and Collision Detections. | ShowRoom (WP)<br><br>TicTacToe (CA)<br><br>PacMan with 2D Array. (WP) | Declare and initialize 2D arrays<br><br>Store and process the game maze using 2D arrays | Students will finalize their Game Projects. |

**List of Problems: Practices and Feedback**

**Project Week: Minimum Requirements:**Player with Health System (Decreasing Health with Time and With Some Enemy Attack). Provision to Store Health Within the System. The player Should have at least two types of weapons (Weapons should be selected from the game). At Least three different types of enemies with Horizontal, Vertical, and Intelligent (Follow the Target) movement. All three Enemies should also have different kinds of weapon systems. Add Scoring System.
**Advanced Requirements:** Two levels with increasing complexity. Loading or some kind of progress bar before the starting of the game. Different colouring scheme of the players and enemies and maze. Player and enemy should be made with more than 1 character (store the player and enemy in separate 2D arrays). Ascii characters to make the players and enemies.

## Projects: Screenshots

### 2D Game

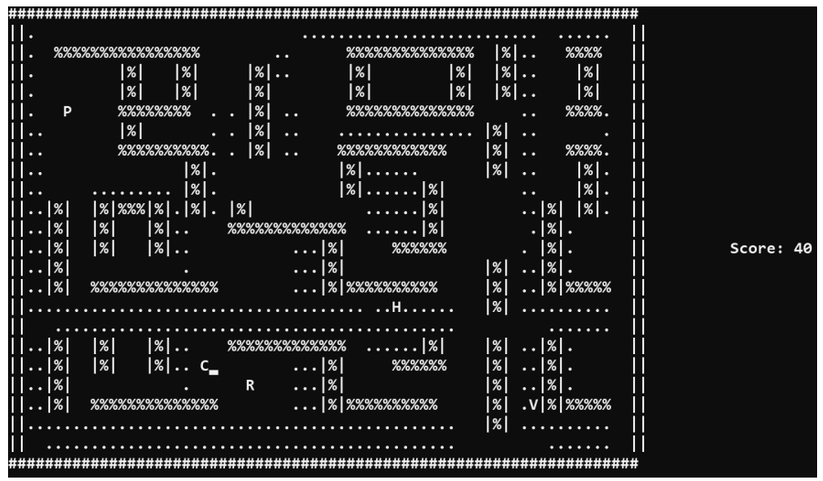


**Figure 1: Game Grid stored in 2D arrays**

**Module 12: Complex Problem Solving and Project ShowCase Week**
**In this module, students will work on complex problems. These complex problems include programming challenges from the top tech companies such as google, facebook, amazon etc. The purpose of this module is to give students confidence and direction to excel their skills so they can join and compete for such international organisations.**

**Big Question:** How to apply computational thinking to solve complex problems ?

| Primary Questions (Why of Module) | Contents (Construct Knowledge) | Engagement (Active Learning) | Hands-on Experience (Skills) | Big Picture Context |
|---|---|---|---|---|
| How to solve complex problems?<br><br>How to effectively present the work ? | **Computationally thinking (CT)**<br>Making students conscious about what we learn so far regarding CT<br>Sequence.<br>Abstraction.<br>Decomposition.<br>Pattern Recognition.<br>Algorithms.<br>Application of Computational Thinking on Complex Problems.<br>**Presentation and Poster Guidelines** | Tournament Winner. (CA)<br>Smallest Difference. (CA)<br>Merge Overlapping intervals. (CA) | Apply computational thinking to solve complex problems.<br><br>Present the work through public demonstration and youtube videos. | Students will showcase their projects: Business Application and Game Application. |

**List of Problems: Practices and Feedback**

**Practice Problems:** Four Number Sum, Subarray Sort, Largest Range, Min Rewards, Zigzag Traverse, Apartment Hunting, Calendar Matching, Waterfall Streams, Minimum Area Rectangle, Line Through Points.
**Quiz:** Quiz 4 will be taken and its solution will be discussed.
**Final Presentation and Video URL**

**Module 13: Python Time - Experiencing any other language**

**So far students will have a good understanding of C++. They have learned two major things 1) Syntax of language 2) Problem Solving Skills. The 2nd part of learning is more difficult and very much common across the different languages. However, the first part varies from language to language as per its design and capabilities but fortunately primary concepts like variables, loop, conditions, array are the same. The only difference is the way to give the commands. For example, in urdu we call it pani but in english we call it water; the need and purpose of both water and pani is the same. Similarly it is for computer languages. Students will see how easily they will boot up in this language.**

**Big Question:** How to learn a new computer language while we know some other language?

| Primary Questions (Why of Module) | Contents (Construct Knowledge) | Engagement (Active Learning) | Hands-on Experience (Skills) | Big Picture Context |
|---|---|---|---|---|
| What is the difference between C++ and Python?<br><br>Why do we require more than one language? | **Python Basics:**<br>Interpreter vs Compiler, Python Installations, First Program, Variables in Python, Conditional Statements-loops and Array in Python.<br>**Advance Python**<br>Functions, File System and Business Application, Defining Functions<br>**More Data Structures.**<br>List, Tuple, Set and Dictionaries<br>Lists, Tuples, Sets, Dictionaries<br>Looping Techniques in Python | Calculator with Python (CA)<br>Addition Function (CL)<br>Reading and Writing into Files<br>SignIn and SignUp Application (CL)<br>Passed Or Failed (CL)<br>Loops<br>Printing Jack (CL)<br>Keep Adding until -1 (CL)<br>Arrays<br>Find the Largest (CL)<br>Pass Fail (WP)<br>Finding largest of given numbers (WP)<br>Login/Signup App with the Python (CA) | Develop a business application using python<br><br>Students should be able to install Python and set up Visual Studio Code on their machines.<br><br>Students should be able to differentiate between Python and other Languages<br><br>Students should be able to write programs to solve complex problems in Python | Students will convert Business Applications in python.<br><br>UAMS with Python |

**List of Problems: Practices and Feedback**

**Project:** Students will make their projects with permanent data storage using python

| General Rubrics for Programming Labs | | | | |
|---|---|---|---|---|
| | Exceptional | Good | Fair | Unsatisfactory |
| | 4 | 3 | 2 | 1 |
| **Problem Solving Approach**<br><br>Understands the problem and its requirements and devise suitable algorithm. And can identify which programming construct/ data structures/ techniques are required as per course requirement | Correct algorithm has been developed and it caters for boundary conditions. Appropriate techniques is employed to do that | Algorithm is correct for major work flow with minor flaws due some boundary conditions or special cases | Algorithm is correct for major work flow but no consideration is given to boundary cases | No understanding of program logic |
| **Design**<br><br>Program should be properly decomposed in reusable components. That either be functions , classes or files or or any other paradigm as per the course requirement | Functionalities are divided properly in coherent and cohesive components | Functionalities are divided into proper coherent units but they are either redundant or lack cohesion | Code is divided into modules but no consideration is put into reusability and cohesion of the modules | No such division of responsibility is visible in the code structure |
| **Completeness**<br><br>Successfully coded the algorithms to meet all the functionality specification of the problem statement | All the functionality requirements are implemented | Most of major functionalities are implemented | At least half of the functionalities of the program are implemented | Very minimal number of minor functionality is implemented |
| **Standard/Clarity**<br><br>Program should be readable, properly commented and follows the standard coding practices | Code is clear , readable, commented and follow standard coding practices | Code is readable but no standard practices are followed | Code is only understandable to one who knows the purpose of it | Code is ambiguous, follows no coding standards |
| **Execution**<br><br>Code is correct, the required programming techniques are implemented accurately according to rules of language. | No Errors, programs compiles and executes perfectly and efficiently | Program does compiles but could have been coded in more efficient way | Program does not compiles have minor errors due to missing semicolons or mis- alignments or missing brackets or any such issue | Program does not compile or interpret due to lack of syntax knowledge |
| **Testing**<br><br>Program executes and all scenarios are tested with no logical errors | All test cases are clear for functionalities and their boundary conditions | All test cases are clear for functionalities but might show erroneous behaviour on boundary conditions | Majority of the test cases are clear, but there might be few failed ones | Majority test cases are failed |

| Rubrics: EVS-BL-R-01 | | | | |
|---|---|---|---|---|
| | A-Extensive Evidence | B-Convincing Evidence | C-Limited Evidence | D-No Evidence |
| Documentation Formatting | All the documentation meets all the criteria. | Documentation is well formatted but some of the criteria is not fulfilled. | Documentation requires a lot of improvement. | Documentation is not Available |
| Documentation Formatting Criteria: In Binder, Title Page, Header-Footers, Font Style, Font Size are all consistent and according to given guidelines. Project Poster is professionally design and well presented | | | | |
| Documentation Contents | Documentation includes all of the criteria. | Documentation meets more than 80% of the criteria given. | Documentation meets more than 50% of the criteria. | When the documentation meets less than 50% of the criteria. |
| Documentation Contents Criteria: Title Page - Table of Contents - Project Abstract - Functional Requirements - Wireframes -Data Flow Diagram-Data Structure (Arrays)-Function Headers and Description - Algorithms and Flowcharts of all functions- Test Cases are defined -Project Code. - Weakness in the Project and Future Directions. - Conclusion and What you Learn from the Project and Course and What is your Future Planning. | | | | |
| Project Complexity | Project has at least 2 user's types and each user has at least 5 functionalities. | Project complexity meet 80% criteria given in extensive evidence | Project complexity meet 50% criteria given in extensive evidence | Project complexity meet less than 50% criteria given in extensive evidence |
| Code Style | All Code style criteria is followed | All code style criteria followed but some improvements required | A lot of improvements required in coding style. | Did not follow code style, |
| Code Style Criteria: Consistent code style. Code is well indented. Variable and Function names are well defined. White Spaces are well used. Comments are added. | | | | |
| Code Documentation Mapping | Code and documentation is synchronized. | Code and documentation does not synchronized at some places | Code and documentation does not synchronized at many places | Code and documentation is not synchronized. |
| Data Structure (Arrays) | Data structure is sufficient for the project requirements | Data Structure is sufficient but require improvement | Data structure is not sufficient and need a lot of improvement | Data Structure is not properly identified and declared. |
| Sorting Features | Sort working 100% and generating useful report | Sorting Feature is working but sorted data is not useful for project. | Sorting feature is partial implemented | Project do not contain sorting |
| Modularity | Meet all Modularity criteria | Meet all Modularity criteria but at some places it is missing | Do not sufficiently meet the modularity criteria. | No modularity or very minimum modularity. |
| Modularity criteria: Functions are defined for each major feature. Functions are independent (identify from parameter list and return types)- Demo Data Functionality Added-At least Two Unit Tests are defined. | | | | |
| Validations | Validations on all number type inputs are applied | Validations are applied but at some places it is missing. | Validations are missing at lot of places | No Validations are used |
| Recommendation Feature | Proper meaning full recommendation is present into system | Partial Recommendation is implemented | Implemented but not meaningful. | Not implemented |
| Presentation and Demo | Presentation and Demo was 100% working | Presentation and Demo require some improvements | Presentation and Demo require a lot of improvements | Presentation was not ok and Demo was not working |

| Rubrics: EVS-GM-R-01 | | | | |
|---|---|---|---|---|
| | A-Extensive Evidence | B-Convincing Evidence | C-Limited Evidence | D-No Evidence |
| Documentation Formatting | All the documentation meets all the criteria. | Documentation is well formatted but some of the criteria is not fulfilled. | Documentation requires a lot of improvement. | Documentation is not Available |
| Documentation Formatting Criteria: In Binder, Title Page, Header-Footers, Font Style, Font Size all are all consistent and according to given guidelines. Project Poster is professionally design and well presented | | | | |
| Documentation Contents | Documentation includes all of the criteria. | Documentation meets more than 80% of the criteria given. | Documentation meets more than 50% of the criteria. | When the documentation meets less than 50% of the criteria. |
| Documentation Contents Criteria: Title Page - Table of Contents - Project Short Description and Story Writing of Game - Game Characters Description - Rules & Interactions - Goal of the Game - Screenshot of the Game - Data Structures Used in the Game - Functions Prototype - Full Code | | | | |
| Project Complexity | Project has at least 1 Player and 3 enemies. Proper use of gotoxy() function. Health system, Firing System and lives decreasing system. | Project complexity meet 80% criteria given in extensive evidence | Project complexity meet 50% criteria given in extensive evidence | Project complexity meet less than 50% criteria given in extensive evidence |
| Randomness | Objects are produced randomly in the game. | Meet more than 80% of the criteria. | Meet more than 50% of the criteria given. | Objects are appearing in the same pattern |
| Code Style | All Code style criteria is followed | All code style criteria followed but some improvements required | A lot of improvements required in coding style. | Did not follow code style, |
| Code Style Criteria: Consistent code style. Code is well indented. Variable and Function names are well defined. White Spaces are well used. Comments are added. | | | | |
| Code Documentation Mapping | Code and documentation is synchronized. | Code and documentation does not synchronized at some places | Code and documentation does not synchronized at many places | Code and documentation does not synchronized. |
| Idea Novelty and Creativity | Idea is unique of the game | Idea is merged by combining other different games | Same idea as a previous game | Could not implement the existing game idea. |
| Data Structure (2D Arrays) | Data structure is sufficient for the project requirements | Data Structure is sufficient but require improvement to meet project requirements. | Data structure is not sufficient and need a lot of improvement | Data Structure is not properly identified and declared. |
| File Handling | Game maze is loaded and the updated maze is stored in the file | Game maze is loaded and partial data is stored in the file. | Game maze is just loaded but the updated game configuration is not stored in the maze. | Project do not contain file handling |
| Modularity | Meet all Modularity criteria | Meet all Modularity criteria but at some places it is missing | Do not sufficiently meet the modularity criteria. | No modularity or very minimum modularity. |
| Modularity criteria: Functions are defined for each major feature. Functions are independent (identify from parameter list and return types)- There is no global variable defined. Arrays and variables are passed as parameters to the functions. Functions exhibit single responsibility principle. | | | | |
| Screen flickering | There is no Screen flickering. | Maze is not flickering but the characters are flickering at great speed | Flickering is done at lot of places | Screen is flickering at all places |
| Presentation/Demo | Presentation and Demo was 100% working | Presentation and Demo require some improvements | Presentation and Demo require improvements | Presentation and Demo was not working |

| Teacher Assistants: Senior students that helped the juniors in the lab tasks to handle difficulties and resolve the errors | | | |
|---|---|---|---|
| Student Roll Number | Student Name | Student Roll Number | Student Name |
| 2020-CS-03 | UMAIR AHMAD | 2020-CS-65 | AHTISHAM UL HAQ |
| 2020-CS-22 | ROHA KHAN | 2020-CS-92 | MUHAMMAD AQIB SHAHZAD |
| 2020-CS-113 | MUHAMMAD ZESHAN AYUB | 2020-CS-75 | MUHAMMAD KHUZAIMA UMAIR |
| 2020-CS-36 | TAJAMMAL MAQBOOL | 2020-CS-52 | MUHAMMAD ABDULLAH UPPAL |
| 2020-CS-114 | MUHAMMAD ALI MURTAZA | 2020-CS-11 | SHANZA |
| 2020-CS-81 | MUHAMMAD ABDULLAH BUTT | 2020-CS-45 | MUHAMMAD FARRUKH HAIDER |
| 2020-CS-72 | ABDULLAH YAQUB | 2020-CS-10 | ASAD MEHMOOD |
| 2020-CS-13 | MUHAMMAD SAFEEULLAH | 2020-CS-26 | MAHNOOR KHAN |
| 2020-CS-40 | MUHAMMAD HAMMAD | 2020-CS-23 | MUHAMMAD RIZWAN |
| 2020-CS-150 | HAFSA RASHID | 2020-CS-19 | MOIZ-UL-HAQ |
| 2020-CS-42 | HASNAT AHMED | 2020-CS-27 | KASHIR SAEED |
| 2020-CS-62 | MUHAMMAD AWAIS YOUSAF BUTT | 2020-CS-57 | MIAN ZAIN UL TAHIR |

| Success Stories: Students that are now earning in the Computing Industry using their learnt skills. | | | |
|---|---|---|---|
| Student Roll Number | Student Name | Job Description | Student Profile Link |
| 2020-CS-46 | Musawar Ahmed Butt | Full-Stack web developer | https://www.fiverr.com/musawir__ahmed?source=gig_page |
| 2020-CS-19 | Moiz-Ul-Haq | Full Stack Developer | https://www.fiverr.com/moiz472 |
| 2020-CS-22 | Roha Khan | Programming Tutor | https://www.fiverr.com/rohakhan04 |
| 2021-CS-189 | Muawwaz Naeem Chishti | Web Developer | https://www.backtoai.com/#team |
| 2021-CS-176 | Azman Shakir | Full Stack Developer | https://www.freeuniversityprojects.com/?m=1 |
| 2021-CS-132, 2021-CS-114 | Muhammad Farman, Muhammad Abu Huraira | Full-Stack Developer | https://github.com/MUHAMMAD-FARMAN/EntryTestWebsite.git |
| 2021-CS-143 | Muhammad Waqas Rashid | C# Developer | https://www.fiverr.com/waqas_aly07?up_rollout=true |
| 2021-CS-149 | Jawad Haider | Work On vue, js and node | Freelance on Fiverr |
| 2021-CS-66 | Asad ullah Khan | Mentor (Programmer) | Paragon Academy (Internship) |
| 2021-CS-65 | Ammar Farooq | Software Developer | Magnatec Systems Private Limited (Internship) |
| 2021-CS-82 | Muhammad Abdullah | Frontend Web Developer | Freelance on Fiverr |
| 2021-CS-90 | Abdullah Javed | Photoshop Editor | https://www.fiverr.com/abdiedits |
| 2021-CS-73 | Talha Ijlal | Junior Web Developer | Tecfare (Part-time) |
| 2021-CS-62 | Syed Abdul Rehman | Linux Kernel Driver Developer | Ebryx (Pvt.) Ltd. (Internship) |
| 2021-CS-125 | Muhammad Ahmad Fraz | Photoshop Editor | https://www.upwork.com/freelancers/~01338b505c1f6cfcee |

| List of Competencies | | |
|---|---|---|
| Competencies | Knowledge Elements | Bloom's Taxonomy Skill Level |
| Explain the Hardware Components (IO Devices, CPU, memory) and their Role while Computing any Problem. | Architecture and Organization<br>• IPO Cycle, Memory, CPU, Fetch Decode Cycle | Understand |
| Explain the Instruction Code, Computational Steps and Algorithm | Architecture and Organization<br>• IPO Cycle, Memory, CPU, Fetch Decode Cycle<br>Programming Fundamentals<br>• Algorithms | Understand |
| Define the Binary Language and its Relationship with Algorithm and Program | Programming Fundamentals<br>• Algorithms, Program | Understand |
| Explain Machine Language, High Level Languages and Role of Compiler | Programming Fundamentals<br>• Algorithms, Program | Understand |
| Define what is Computational Thinking and its four pillars | Analytical and Critical Thinking | Understand |
| Write an Algorithm that converts Input into the required Output using Variables, Constants and Arithmetic Operators. | Programming Fundamentals<br>• Variables, Expressions, Arithmetic operators | Apply |
| Evaluate the Correctness of Algorithms with Different Test Cases. | Programming Fundamentals<br>• Algorithms<br>Testing and debugging | Understand |
| Write and execute a computer program that shows output on monitor screen (Console) | Programming Fundamentals<br>• Output on Console | Apply |
| Explain why we need Variables and what is their Relation with the Memory. | Programming Fundamentals<br>• Variables<br>Architecture and Organization<br>• Memory | Remember |

| | | |
|---|---|---|
| Explain what is a Data Type, why we need it and what is its Role in Variable Declaration | Programming Fundamentals<br>• Variables, Datatypes | Remember |
| Write expressions using Variables, Constants and Arithmetic Operators | Programming Fundamentals<br>• Variables, Expressions, Arithmetic operators | Apply |
| Write Expressions while considering the Operator Precedence Rules | Programming Fundamentals<br>• Variables, Expressions, Arithmetic operators, Precedence Rule | Apply |
| Write a Program in C++ that Declares Variable, Stores Value in it and Prints its Value on Console | Programming Fundamentals<br>• Variables, Output on Console | Apply |
| Write a C++ program that evaluates expressions consisting of Arithmetic Operators, Constants and Variables | Programming Fundamentals<br>• Variables, Expressions, Arithmetic operators, Output on Console | Apply |
| Write a C++ program that takes input from the user, applies mathematical operations on it and then converts that input into output. | Programming Fundamentals<br>• Input, Variables, Expressions, Arithmetic operators, Output on Console | Apply |
| Write a C++ program using a single conditional statement with one Boolean expression consisting of an Equal comparison operator. (IF Statement) | Programming Fundamentals<br>• IF statement, Comparison Operators | Apply |
| Write a C++ program using a single conditional statement with one Boolean expression consisting of Any comparison operator. (IF Statement) | Programming Fundamentals<br>• IF statement, Comparison Operators | Apply |
| Write a C++ program that involves multiple decision making using conditional statements. (Multiple IF Statements) | Programming Fundamentals<br>• Multiple IF statements, Comparison Operators | Apply |
| Write a C++ program that makes multiple decision making more optimized by checking fewer conditional statements. (IF-Else Statement) | Programming Fundamentals<br>• IF-Else statements, Comparison Operators | Apply |
| Write a C++ program that makes complex decision making using nested conditional statements. (Nested IF Statement) | Programming Fundamentals<br>• Nested IF statements, Comparison Operators | Apply |

| | | |
|---|---|---|
| Write a C++ program that involves complex decision making using a single conditional statement with two Boolean expressions and the AND logical operator. | Programming Fundamentals<br>• Conditional statements, Comparison Operators, Logical Operators | Apply |
| Write a C++ program that involves complex decision making using a single conditional statement with two Boolean expressions and the OR logical operator. | Programming Fundamentals<br>• Conditional statements, Comparison Operators, Logical Operators | Apply |
| Write a C++ program that involves complex decision making using a single conditional statement with two Boolean expressions and the NOT logical operator. | Programming Fundamentals<br>• Conditional statements, Comparison Operators, Logical Operators | Apply |
| Write a C++ program that involves complex decision making using a single conditional statement with multiple Boolean expressions and multiple logical operators. | Programming Fundamentals<br>• Conditional statements, Comparison Operators, Logical Operators | Apply |
| Write a C++ program that involves complex decision making using a single conditional statement with multiple Boolean expressions and multiple logical operators while considering the precedence rules. | Programming Fundamentals<br>• Conditional statements, Comparison Operators, Logical Operators, Precedence Rules | Apply |
| Write a C++ Program that repeats a Set of Instructions for a specific number of times to solve the given problem using Counter Loop. | Programming Fundamentals<br>• Counter Loop (For Loop) | Apply |
| Write a C++ Program that repeats a Set of Instructions for an unknown number of times to solve the given problem using Conditional Loop. | Programming Fundamentals<br>• Conditional Loop (While Loop) | Apply |
| Write a C++ Program that repeats a Set of Instructions for any number of times to solve the given problem using Counter Loop. | Programming Fundamentals<br>• Counter Loop (For Loop) | Apply |
| Write a C++ Program that solves larger problems by decomposing it into smaller subproblems and combining their solution to achieve final results using the Counter Loop. | Programming Fundamentals<br>• Computational thinking, Counter Loop (For Loop) | Apply |

| | | |
|---|---|---|
| Write a C++ Program that repeats a complex set of instructions using nested loops. | Programming Fundamentals<br>• Counter Loop (For Loop), Nested Loops | Apply |
| Write a program that alters the normal flow of the loop using continue and break statements. | Programming Fundamentals<br>• Counter Loop (For Loop), Continue Statement, Break Statement | Apply |
| Identify the scenarios when scalar variables are not sufficient to handle the data and subsequently arrays are required. | Programming Fundamentals<br>• Limitation of variables | Analyze |
| Write a C++ program that declares an Array, takes a large number of elements as input from the user in that array and then retrieves that data from the array | Programming Fundamentals<br>• Arrays, CRUD operations on Arrays | Apply |
| Write a C++ program that declares an Array, takes an unknown number of elements as input from the user and then retrieves that data from the array | Programming Fundamentals<br>• Arrays, CRUD operations on Arrays | Apply |
| Write a C++ program that processes data elements in the array and make them in an order. | Programming Fundamentals<br>• Arrays, CRUD operations on Arrays | Apply |
| Write a C++ program for storing and processing multiple information of a record using parallel Arrays. | Programming Fundamentals<br>• Parallel Arrays, CRUD operations on Parallel Arrays | Apply |
| Write a Code that requires lengthy and repeated structure to solve complex problems. | Programming Fundamentals<br>Analytical and Critical Thinking | Apply |
| Identify the problems that arise due to the lengthy and repeated code for complex problems | Programming Fundamentals<br>Analytical and Critical Thinking | Analyze |
| Write a program that calls predefined functions to generate the results. | Programming Fundamentals<br>• Functions (Pre-defined) | Apply |
| Write a program that calls a user-defined function to generate the results | Programming Fundamentals<br>• Functions (User-defined) | Apply |

| | | |
|---|---|---|
| Write a user defined function to solve the code repetition issue. | Programming Fundamentals<br>• Functions (User-defined)<br>Analytical and Critical Thinking | Apply |
| Understand the importance of function prototype. | Programming Fundamentals<br>• Functions (User-defined), Function Prototype | Remember |
| Write a code that uses global variables for communicating between functions. | Programming Fundamentals<br>• Functions (User-defined), Function Prototype, Global variables | Apply |
| Identify the difference between High coupled and Low coupled functions. | Programming Fundamentals<br>• Functions (User-defined), High VS Low coupled Functions | Analyze |
| Write a highly cohesive and low coupled function that can be reused in complex problems. | Programming Fundamentals<br>• Functions (User-defined), Function Prototype, Global variables<br>Analytical and Critical Thinking | Apply |
| Write a Code to show the limitations of taking input and then displaying the output on the console | Programming Fundamentals<br>• File Handling<br>Analytical and Critical Thinking | Apply |
| Write a code that solves the limitations of taking input from the user | Programming Fundamentals<br>• File Handling<br>Analytical and Critical Thinking | Apply |
| Write a code that reads character by character from the file. | Programming Fundamentals<br>• File Handling, CRUD operations using files | Apply |
| Write a code that creates (store) data into the permanent storage | Programming Fundamentals<br>• File Handling, CRUD operations using files | Apply |
| Write a code that appends (insert) data into the permanent storage. | Programming Fundamentals<br>• File Handling, CRUD operations using files | Apply |
| Write a code that searches from the formatted (comma separated file) storage. | Programming Fundamentals<br>• File Handling, CRUD operations using files | Apply |

| | | |
|---|---|---|
| Development of data driven applications with permanent storage. | Programming Fundamentals<br>• File Handling<br>Analytical and Critical Thinking | Create |
| Write a program that stores similar records using two dimensional arrays. | Programming Fundamentals<br>• 2D Arrays | Apply |
| Write a story for the 2D tile-based game, identify the characters (Player and enemies), game rules, goals, interactions and reward system | Analytical and Critical Thinking | Remember |
| Write a program that stores game maps and game objects into the 2D Array and show the map with the objects on the console from the 2D Array | Programming Fundamentals<br>• 2D Arrays | Apply |
| Write a program to move a game object on the console using arrow keys. | Programming Fundamentals<br>• Key Strokes<br>Analytical and Critical Thinking | Apply |
| Write a program to detect interaction between objects and update the game state accordingly. | Programming Fundamentals<br>• 2D Arrays<br>Analytical and Critical Thinking | Apply |
| Write a program that stores data of a 2D array into file and loads the data back to 2D Array using file handling. | Programming Fundamentals<br>• 2D Arrays, File Handling<br>Analytical and Critical Thinking | Apply |
| Write a program that saves the Game Map into the file and loads it back to the 2D array. | Programming Fundamentals<br>• 2D Arrays, File Handling<br>Analytical and Critical Thinking | Apply |
| Develop the Pac-Man game by loading data from the file. | Programming Fundamentals<br>• 2D Arrays, File Handling<br>Analytical and Critical Thinking | Create |

**References:**

Brophy, J. E. (2013). Motivating students to learn. Routledge. Caro-Alvaro, S., Garcia-Lopez, E., Garcia-Cabot, A., De-Marcos, L., & Martinez-Herriaz, J.-J.

Mekler, E. D., Brühlmann, F., Tuch, A. N., & Opwis, K. (2017). Towards understanding the effects of individual gamification elements on intrinsic motivation and performance. Computers in Human Behaviour, 71, 525–534.

Hattie, J. (2009). Visible learning, Oxford, UK: Routledge, p173

(Sarpong, 2013). Sarpong, Kofi Adu-Manu, John Kingsley Arthur, and Prince Yaw Owusu Amoako. "Causes of failure of students in computer programming courses: The teacher-learner Perspective." *International Journal of Computer Applications* 77, no. 12 (2013).

(A. Robins, 2003). A. Robins, J. Rountree, and N. Rountree, Learning and teaching programming: A review and discussion, Comput. Sci. Educ. 13 (2003), no. 2, 137–172.

(Kiesler,2020). Kiesler, Natalie. "Towards a Competence Model for the Novice Programmer Using Bloom's Revised Taxonomy-An Empirical Approach." In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, pp. 459-465. 2020.