“ELECTRONIC VOTING MACHINE”

A project report submitted in the partial fulfilment the award of degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

By

Maddula Naga Sai Preetham	- 221801350007
Maddula Raja Prem Sai	- 221801390008
Ch Rohith Vinay	- 221801350017
D Bhaskar Viswanadh	- 221801350019
S Sai Kiran	- 221801350002
T Jai Sandesh	- 221801390003

Under the esteemed Guidance of

Mr. B. Lakshmana Rao,

MTech,

Asst.Professor, CUTMAP

Centurion University of Technology and Management

Vizianagaram Pin: 535003, A.P, India (2022-2023)

Centurion University of Technology and Management

Vizianagaram Pin: 535003, A.P, India

(2022-2023)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that the project work entitled “**ELECTRON VOTING MACHINE”** is a fulfillment of project work done by **Naga Sai Preetham (221801350007), Raja Prem Sai (221801390008), Rohith Vinay (221801350017), Bhaskar Viswanadh (221801350019), Sai Kiran (221801350002), Jai Sandesh (221801390003)** for the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING, Centurion University of Technology and Management**, during the academic year 2022-2023.

INTERNAL GUIDE
Mr. B. Lakshman Rao
Asst.Professor

HEAD OF THE DEPARTMENT
Dr A. Sateesh
Associate Professor

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

It is with at most pleasure and excitement we submit our project partial fulfillment of the requirement for the Bachelor of Technology.

The project is a result to the cumulative efforts, support, guidance, encouragement and inspiration from many of those for whom we have to give our truthful honor and express gratitude through bringing out this project at the outset as per our knowledge.

We convey special thanks to our project **Guide B Lakshman Rao, MTech** who has guided us and encouraged us to enhance our knowledge with present working of this project to bring out enriching the quality of project.

We express our graditute to **Dr A Sateesh, MTech, Asst.Professor and Head of the Department**, who facilitated us to providing the friendly environment which helped to enhance our skills in present project.

DECLARATION

We hereby declare that the project entitled "**ELECTRON VOTING MACHINE**" submitted to the fulfillment the degree of **B. TECH (CSE)** in **Centurion University of Technology and Management.** This project work in original has not been submitted so far in any part or full for any other university or institute for the award of any degree.

Maddula Naga Sai Preetham	221801350007
Maddula Raja Prem Sai	221801390008
Ch Rohith Vinay	221801350017
D Bhaskar Viswanadh	221801350019
S Sai Kiran	221801350002
T Jai Sandesh	221801390003

ABSTRACT

Electronic Voting Machine (EVM) is a simple electronic device used to record votes in place of ballot papers and boxes which were used earlier in conventional voting system. All earlier elections be it state elections or central elections a voter used to cast his/her favorite candidate by putting the stamp against his/her name and then folding the ballot paper as per a prescribed method before putting it in the Ballot Box. This is a long, time-consuming process and very much prone to errors. This situation continued till election scene was completely changed by electronic voting machine. No more ballot paper, ballot boxes, stamping, etc. all this condensed into a simple box called ballot unit of the electronic voting machine. Because EVM cannot be easily misplaced, forged, or shared, they are considered more reliable for person recognition than traditional token or knowledge-based methods. So, the electronic voting system has to be improved based on the current technologies. This project discusses the Electronic Voting Machine by using the software program in C.

CONTENTS

INTRODUCTION TO EVM

Electronic Voting Machine is a simple electronic device used to record votes in place of ballot papers and boxes which were used earlier in conventional voting system. It is a simple machine that can be operated easily by both the polling personnel and the voters. Being a stand alone machine without any network connectivity, nobody can interfere with its programming and manipulate the result. Keeping the erratic power supply position in many places in the country, the machines have been made to run on batteries. It has mainly two units: Control unit and Ballot unit. The Control Unit is the main unit which stores all data and controls the functioning of EVM. The program which controls the functioning of the control unit is burnt into a microchip on a "one time programmable basis". Once burnt it cannot be read, copied out or altered. The EVMs use dynamic coding to enhance security of data transmitted from ballot unit to control unit. The new EVMs have also got real time clock and date-time stamping facility which enables them to record the exact time and date whenever a key is pressed. After the voting is completed and the close button is pressed, the machine does not accept any data or record any vote. Through the press of "total" button, the control unit can display the number of votes recorded till that time which can be cross checked with the register of voters. The display system of the control unit shows the total number of votes polled in a polling station and the candidate-wise votes polled in the machine when the 'result' button is pressed by the counting staff in the presence of counting agents at the counting centre. The control unit can also detect any physical tampering made with the connecting cable and indicate the same in the display unit.

In previous manual elections in India, a nationwide ballot could consume around 8,000 tonnes of paper and 400,000 phials of indelible ink and require some 2.5 million strongboxes to store them under heavy security until the votes were counted. In the past, it took up to three – four days to count the votes, with hired personnel spending day and night in secured areas manually counting each ballot. Sometimes demanding for recounting resulting for the low margin of difference of votes between the top two candidates coupled with large number of invalid and doubtful votes.

SYSTEM ANALYSIS

The procedure of Election with straightforwardness and security is basic in any nation to guarantee the privilege of voting in favor of the residents. As misleading is an unavoidable danger to voting, it is key that an electronic voting structure gives an anomalous condition of security. Currently available Electronic Voting Machines in India have been applauded for their straightforward configuration, usability, and dependability, yet as of late they have additionally been condemned taking after across the board reports of race abnormalities. Notwithstanding this feedback, numerous subtle elements of the machines' outline have never been freely unveiled, and they have not been subjected to a thorough, autonomous security assessment. We know the entire world effectively dismisses the electronic voting machine and just a couple of nations are currently utilizing the electronic voting machines. The paper gives an establishment in planning secure and down to earth voting plans to create a protected, productive and freely adequate execution of voting process in this present reality.

EXISTING SYSTEM (Ballot System)

A **ballot** is a device used to cast votes in an election and may be found as a piece of paper or a small ball used in secret voting. It was originally a small ball (see blackballing) used to record decisions made by voters in Italy around the 16th century.

Each voter uses one ballot, and ballots are not shared. In the simplest elections, a ballot may be a simple scrap of paper on which each voter writes in the name of a candidate, but governmental elections use pre-printed ballots to protect the secrecy of the votes. The voter casts their ballot in a box at a polling station.

In British English, this is usually called a "ballot paper". The word *ballot* is used for an election process within an organization (such as a trade union "holding a ballot" of its members).

DISADVANTAGES

Paper ballots are considered to be the most trustworthy methods to be used during an election but it comes with few limitations. We have listed down few disadvantages of the paper ballots due to which EVMs are preferred nowadays.

1. There is no scope for automation in paper ballot system. Electronic voting machines are preferred because it reduces the manual work and with one press of the button, the votes are recorded.

2. Post-election, it takes a huge amount of time to count the votes before declaring the results. In electronic voting machines, the counting is be done within few minutes.

3. The people who are physically challenged find it difficult to cast their votes through the paper ballot and even if they cast their votes using paper ballot they require someone to cast their vote on behalf. In such cases, their privacy while casting vote is breached. However, with EVMs in place they can just touch the screen in order to cast their votes via touchscreen EVM.

4. Paper is a substance that is inflammable thus under certain circumstances, the paper in which the votes were recorded in ballot might get damaged then becomes impossible to retrieve the records of the votes.

5. Paper ballots cannot be manipulated but using paper as a raw material in such voting system it becomes harmful for the environment. On the other hand, using electronic voting machines are much more economical.

6. In few places where the governance is corrupt, they can easily insert several bogus paper votes in the ballot and then it becomes impossible to track the honest votes.

7. Few electronic voting machines give paper trial for the votes recorded. However, on the paper ballot, there are no such confirmations. There is no automation in place which can tell that how many votes were recorded per minute.

8. Casting votes using paper ballot is a time-consuming task whereas voting via EVM is done in few seconds.

9. The cost of expenditure on the paper ballot is way higher than on EVM. Using EVMs are quite economical for the election commission.

10. With the paper ballot, the greatest challenge is that one could never use historic reference in case of the paper ballot whereas in EVMs one can store the records for years.

PROPOSED SYSTEM

Electronic voting (also known as **e-voting**) is voting that uses electronic means to either aid or take care of casting and counting ballots.

Depending on the particular implementation, e-voting may use standalone *electronic voting machines* (also called EVM) or computers connected to the Internet (**online voting**). It may encompass a range of Internet services, from basic transmission of tabulated results to full-function online voting through common connectable household devices. The degree of automation may be limited to marking a paper ballot, or may be a comprehensive system of vote input, vote recording, data encryption and transmission to servers, and consolidation and tabulation of election results.

ADVANTAGES

Advantages for election management bodies

1.Faster counting and delivering of election results.
2.Increased trust in elections as human error is avoided.
3.Increased voter turnout, especially when internet voting is involved.
4.Cost reduction when applying e-Voting on multiple electoral events.
5.Reduced ballot waste.

Improved convenience for voters

1.Easier vote marking and casting as the voting experience helps to avoid errors, in particular when over-voting, under-voting or making incorrect selections.
2.Ballots available in multiple languages.
3.Vote remotely from home or other locations using internet voting.

Improved Accessibility

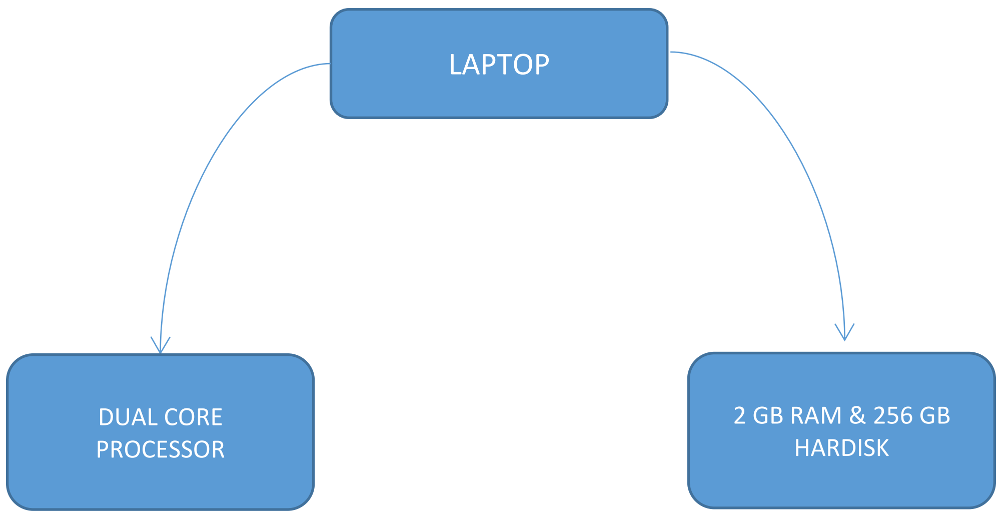
1.People with disabilities are able to vote thanks to features like sip-and-puff voting, paddle voting,high-contrast viewing screens and audio voting.
2.Online voting allows people who are unable to reach the polling places to vote.

Fraud Prevention

1.e-Voting reduces the chances of accidental or intentional variations in vote counts by reducing poll worker direct interaction with ballots or counts.
2.E-Voting and online voting reduce voter errors and the chances of voter fraud,increasing electoral integrity.

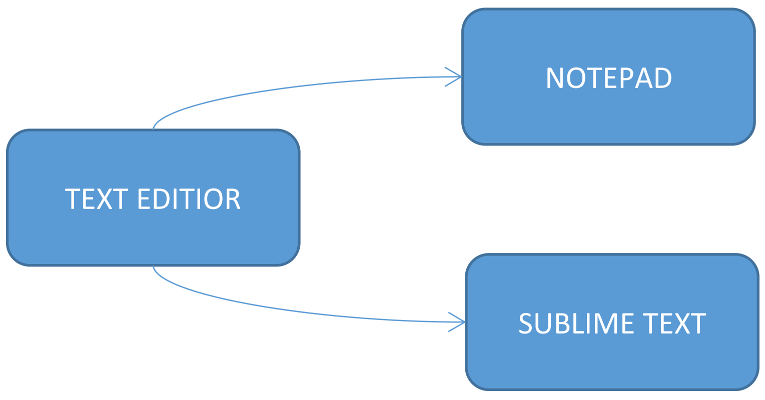
SYSTEM REQUIREMENTS SPECIFICATION

Minimun hardware requirements

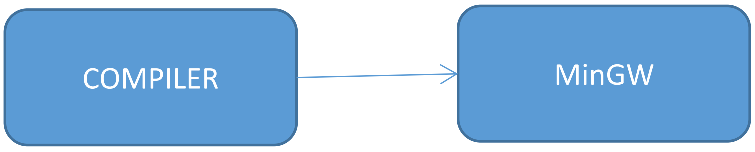


Minimum software requirements

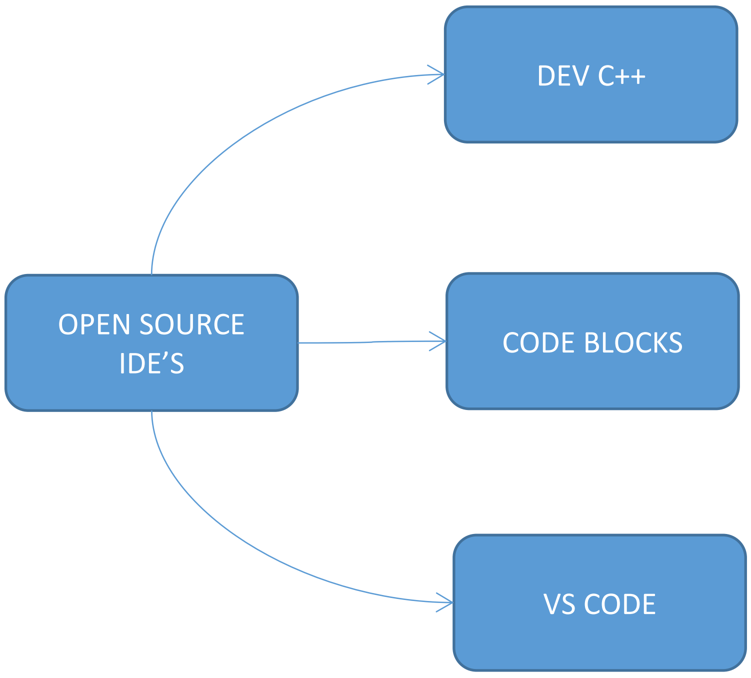
1.



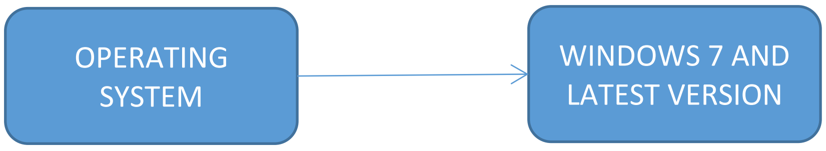
2.


COMPILER
MinGW

3.


OPEN SOURCE IDE’S
DEV C++
CODE BLOCKS
VS CODE

4.


OPERATING SYSTEM
WINDOWS 7 AND LATEST VERSION

CODE DESIGN

This code basically works with the librarys in c language. It works in a 2 way , input and output . The user basically enters the required option and it takes the user to that portal where he has to enter the username and the password, if he had entered wrong username or password, he also gets a option to re-enter the credentials. There are 2 roles one is admin and the voter; it works for both the of them. The program analyse the number of votes that are received and it shows the live voting and the winner. The voter should choose his name in the following list and he has to enter the user-id and password so he can access to his portal. It is a fast and secure way for the voting.

UNIFIDE MODELING LANGUAGE (UML) DIAGRAMS

UML is a general-purpose modeling language. It was initially started to capture the behavior of complex software and non-software system and now it has become an OMG standard.

UML provides elements and components to support the requirement of complex systems. UML follows the object-oriented concepts and methodology. So object oriented systems are generally modeled using the pictorial language.

UML diagrams are drawn from different perspectives like design, implementation, deployment etc.

At the conclusion UML can be defined as a modeling language to capture the architectural, behavioral and structural aspects of a system.

Objects are the key to this object-oriented world. The basic requirement of object-oriented analysis and design is to identify the object efficiently. After that the responsibilities are assigned to the objects. Once this task is complete the design is done using the input from analysis.

The UML has an important role in this OO analysis and design, The UML diagrams are used to model the design. So, the UML has an important role to play.

UML notations:

UML notations are the most important elements in modeling. Efficient and appropriate use of notations is very important for making a complete and meaningful model. The model is useless unless its purpose is depicted properly.

So learning notations should be emphasized from the very beginning. Different notations are available for things and relationships. And the UML diagrams are made using the notations of things and relationships. Extensibility is another important feature which makes UML more powerful and flexible.

UML Diagrams:

Diagrams are the heart of UML. These diagrams are broadly categorized as structural and behavioral diagrams.

- Structural diagrams are consisting of static diagrams like class diagram, object diagram etc.
- Behavioral diagrams are consisting of dynamic diagrams like sequence diagram, collaboration diagram etc.

The static and dynamic nature of a system is visualized by using these diagrams.

Class diagrams:

Class diagrams are the most popular UML diagrams used by the object-oriented community. It describes the objects in a system and their relationships. Class diagram consists of attributes and functions.

A single class diagram describes a specific aspect of the system and the collection of class diagrams represents the whole system. Basically, the class diagram represents the static view of a system.

Class diagrams are the only UML diagrams which can be mapped directly with object-oriented languages. So, it is widely used by the developer community.

Object Diagram:

An object diagram is an instance of a class diagram. So, the basic elements are similar to a classdiagram. Object diagrams are consisting of objects and links. It captures the instance of the system at a particular moment.

Object diagrams are used for prototyping, reverse engineering and modeling practical scenarios.

Component Diagram:

Component diagrams are special kind of UML diagram to describe static implementation view of a system. Component diagrams consist of physical components like libraries, files, folders etc.

This diagram is used from implementation perspective. More than one component diagrams are used to represent the entire system. Forward and reverse engineering techniques are used to make executables from component diagrams.

Deployment Diagram:

Component diagrams are used to describe the static deployment view of a system. These diagrams are mainly used by system engineers.

Deployment diagrams are consists of nodes and their relationships. An efficient deployment diagram is an integral part of software application development.

Use Case Diagram:

Use case diagram is used to capture the dynamic nature of a system. It consists of use cases, actors and their relationships. Use case diagram is used at a high level design to capture the requirements of a system.

So it represents the system functionalities and their flow. Although the use case diagrams are not a good candidate for forward and reverse engineering but still they are used in a slightly differently way to model it.

Interaction Diagram:

Interaction diagrams are used for capturing dynamic nature of a system. Sequence and collaboration diagrams are the interaction diagrams used for this purpose.

Sequence diagrams are used to capture time ordering of message flow and collaboration diagrams are used to understand the structural organization of the system. Generally a set of sequence and collaboration diagrams are used to model an entire system.

Statechart Diagram:

Statechart diagrams are one of the five diagrams used for modeling dynamic nature of a system. These diagrams are used to model the entire life cycle of an object. Activity diagram is a special kind of Statechart diagram.

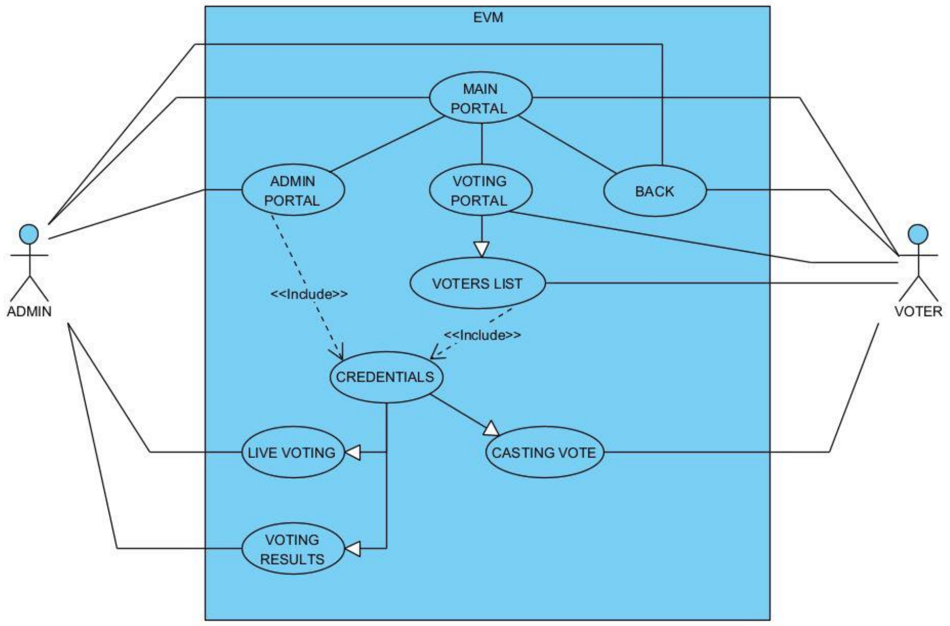
State of an object is defined as the condition where an object resides for a particular time and the object again moves to other states when some events occur. Statechart diagrams are also used for forward and reverse engineering.

Activity Diagram:

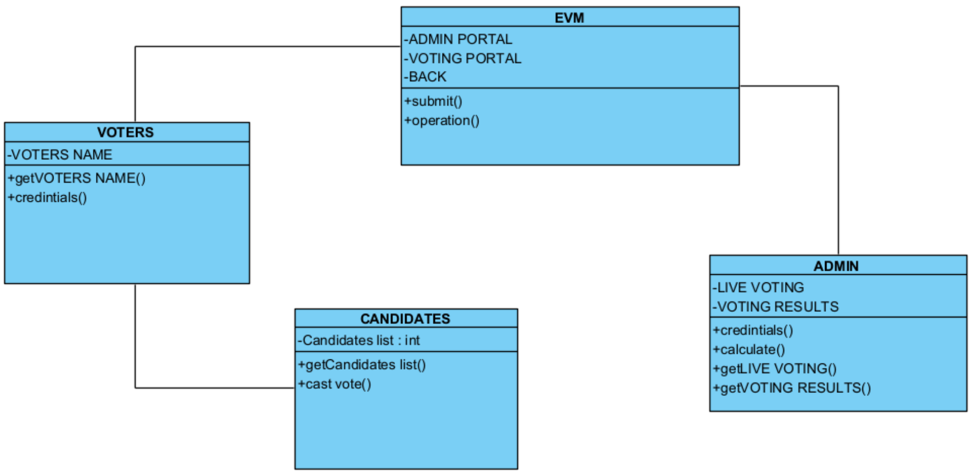
Activity diagram is another important diagram to describe dynamic behaviour. Activity diagram consists of activities, links, relationships etc. It models all types of flows like parallel, single, concurrent etc.

Activity diagram describes the flow control from one activity to another without any messages. These diagrams are used to model high level view of business requirements.

1.USE CASE DIAGRAM



2.CLASS DIAGRAM



IMPLEMENTATION

TECHNOLOGIES USED

Windows 10 Operating System

Microsoft has begun at 1981 with MSDOS 1.0 to develop operating systems for computers. One year ahead Microsoft has worked in cooperation on the Unix derivative operating system XENIX OS for different computer platforms, this OS field however was transferred to SCO in 1984. With Windows 1.0 were added in 1985 beside DOS a second OS line, which was meant first for single workplaces for Consumer (Home edition) and later with added network support.

The third product line was started with MS OS/2 1.0 in 1987. The professional edition was for server applications and network clients designed. In February 1989 the development of Windows NT started (NT = New Technology), the first version was published with Windows NT 3.1 in July 1993. Up to 200 developers had programmed at the same time on the approx. 6 million code lines. While MS-DOS was programmed nearly completely in assembler, Windows NT also consists of source code of the programming language C. Up to 450 developers were involved at the operating system Windows NT 3.51 which was released in May 1995. To record times up to 800 developers worked on the successor Windows NT 4.0 for the release in July 1996.

Windows is a series of operating systems developed by Microsoft. Each version of Windows includes a graphical user interface, with a desktop that allows users to view files and folders in windows. For the past two decades, Windows has been the most widely used operating system for personal computers PCs.

Microsoft Windows is designed for both home computing and professional purposes. Past versions of Windows home editions include Windows 3.0 (1990), Windows 3.1 (1992), Windows 95 (1995), Windows 98 (1998), Windows Me (2000), Windows XP (2001), and Windows Vista (2006). The current version, Windows 7, was released in 2009. The first business-oriented version of Windows, called Windows NT 3.1, was in 1993. This was followed by Windows 3.5, 4.0, and Windows 2000. When Microsoft released Windows XP in 2001, the company simply created different editions of the operating system for personal and business purposes. Windows Vista and Windows 7 have followed the same release strategy.

Windows is designed to run on standard x86 hardware, such as Intel and AMD processors. Therefore, it can be installed on multiple brands of hardware, such as Dell, HP, and Sony computers, as well as home-built PCs. Windows 7 also includes several touchscreen features, that allow the operating system to run on certain tablets and computers with touchscreen displays. Microsoft's mobile operating system, Windows Phone, is designed specifically for smartphones and runs on several brands of phones, including HTC, Nokia, and Samsung.

Windows 2000 was the ambitious project following on this, up to 1.400 developer worked on the 29 million code lines. The development costs amounted to about 1 billion dollar. Altogether 5.000 developers worked on the 50 million code lines of assemblers, C and C++, for the Windows Server 2003 operating system with release in April 2003. The development of operating system versions for the MIPS, PowerPC and alpha architecture became gradually cancelled up to the market release of Windows 2000. This was also involved by the lacking driver and software support for these platforms. With Windows CE 1.0 a new product line for small devices (PDAs) was created in 1996.

The former splitting into Consumer and Business Windows Edition is to be united with Windows XP (alias Whistler) again and continued in this product line. Thus is void for the first time the condition of MSDOS, which need even Windows 95 to ME for the system start. Directly with all Windows versions so far the drive assembly marking with the letters A to Z whereby the maximally manageable number on 26 is limited, exluded mounted network directories.

INTRODUCTION TO C-LANGUAGE

C is a programming language developed at AT & T's Bell Laboratories of USA in 1972. It was designed and written by a man named Dennis Ritchie. In the late seventies C began to replace the more familiar languages of that time like PL/I, ALGOL, etc.

ANSI C standard emerged in the early 1980s, this book was split into two titles: The original was still called ***Programming in C***, and the title that covered ANSI C was called ***Programming in ANSI C***. This was done because it took several years for the compiler vendors to release their ANSI C compilers and for them to become ubiquitous. It was initially designed for programming UNIX operating system. Now the software tool as well as the C compiler is written in C. Major parts of popular operating systems like Windows, UNIX, Linux is still written in C. This is because even today when it comes to performance (speed of execution) nothing beats C. Moreover, if one is to extend the operating system to work with new devices one needs to write device driver programs. These programs are exclusively written in C. Cseems so popular is because it is **reliable**, **simple** and **easy** to use. often heard today is – "C has been already superceded by languages like C++, C# and Java.

C LANGUAGE BASIC SYNTAX

Tokens in C

A C program consists of various tokens and a token is either a keyword, an identifier, a constant, a string literal, or a symbol. For example, the following C statement consists of five tokens:

```
printf("Hello, World! \n");
```

The individual tokens are:

```
printf
(
"Hello, World! \n"
)
;
```

Semicolons ;

In C program, the semicolon is a statement terminator. That is, each individual statement must be ended with a semicolon. It indicates the end of one logical entity.
For example, following are two different statements:

```
printf("Hello, World! \n");
return 0;
```

Comments

Comments are like helping text in your C program and they are ignored by the compiler.
They start with /* and terminates with the characters */ as shown below:

```
/* my first program in C */
```

You cannot have comments within comments and they do not occur within a string or character literals.

Identifiers

A C identifier is a name used to identify a variable, function, or any other user-defined item. An identifier starts with a letter A to Z or a to z or an underscore _ followed by zero or more letters, underscores, and digits (0 to 9).

C does not allow punctuation characters such as @, $, and % within identifiers. C is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in C. Here are some examples of acceptable identifiers:

```
mohd        zara    abc    move_name   a_123
myname50    _temp   j      a23b9       retVal
```

Keywords

The following list shows the reserved words in C. These reserved words may not be used as
constant or variable or any other identifier names.

auto	else	Long	switch
break	enum	register	typedef
case	extern	return	union
char	float	short	unsigned
const	for	signed	void
continue	goto	sizeof	volatile
default	if	static	while
do	int	struct	_packed
double			

White space in C

A line containing only white space, possibly with a comment, is known as a blank line, and a C compiler totally ignores it.

White space is the term used in C to describe blanks, tabs, newline characters and comments. White space separates one part of a statement from another and enables the compiler to identify where one element in a statement, such as int, ends and the next element begins. Therefore, in the following statement:

```
int age;
```

There must be at least one white space character (usually a space) between int and age for the compiler to be able to distinguish them. On the other hand, in the following statement:

```
fruit = apples + oranges;    // get the total fruit
```

No white space characters are necessary between fruit and =, or between = and apples, although you are free to include some if you wish for readability purpose.

C DATA TYPES

In the C programming language, data types refer to an extensive system used for declaring variables or functions of different types. The type of a variable determines how much space it occupies in storage and how the bit pattern stored is interpreted.

The types in C can be classified as follows:

S.N.	Types and Description
1	**Basic Types:** They are arithmetic types and consists of the two types: (a) integer types and (b) floating-point types.
2	**Enumerated types:** They are again arithmetic types and they are used to define variables that can only be assigned certain discrete integer values throughout the program.
3	**The type void:** The type specifier *void* indicates that no value is available.
4	**Derived types:** They include (a) Pointer types, (b) Array types, (c) Structure types, (d) Union types and (e) Function types.

The array types and structure types are referred to collectively as the aggregate types. The type of a function specifies the type of the function's return value. We will see basic types in the following section, whereas, other types will be covered in the upcoming chapters.

Integer Types

Following table gives you details about standard integer types with its storage sizes and value ranges:

Type	Storage size	Value range
Char	1 byte	-128 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127
Int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295
Short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
Long	4 bytes	-2,147,483,648 to 2,147,483,647
unsigned long	4 bytes	0 to 4,294,967,295

To get the exact size of a type or a variable on a particular platform, you can use the **sizeof** operator. The expressions **sizeof(type)** yields the storage size of the object or type in bytes. Following is an example to get the size of **int** type on any machine:

```
#include <stdio.h>
#include <limits.h>

int main()
{
   printf("Storage size for int : %d \n", sizeof(int));

   return 0;
}
```

When you compile and execute the above program, it produces the following result on Linux:

```
Storage size for int : 4
```

Floating-Point Types

Following table gives you details about standard floating-point types with storage sizes and value ranges and their precision:

Type	Storage size	Value range	Precision
float	4 byte	1.2E-38 to 3.4E+38	6 decimal places
double	8 byte	2.3E-308 to 1.7E+308	15 decimal places
long double	10 byte	3.4E-4932 to 1.1E+4932	19 decimal places

The header file **float.h** defines macros that allow you to use these values and other details about the binary representation of real numbers in your programs. Following example will print storage space taken by a float type and its range values:

```
#include <stdio.h>
#include <float.h>

int main()
```

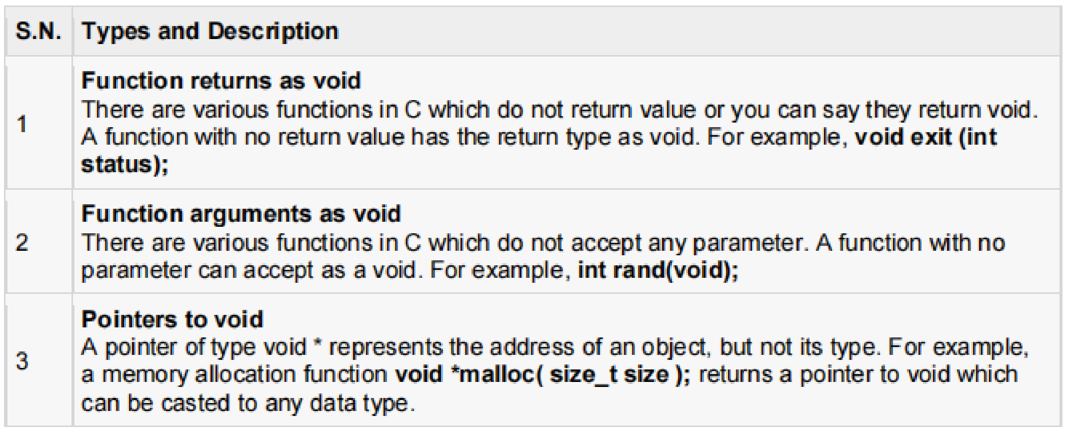
```
{
   printf("Storage size for float : %d \n", sizeof(float));
   printf("Minimum float positive value: %E\n", FLT_MIN );
   printf("Maximum float positive value: %E\n", FLT_MAX );
   printf("Precision value: %d\n", FLT_DIG );

   return 0;
}
```

When you compile and execute the above program, it produces the following result on Linux:

```
Storage size for float : 4

Minimum float positive value: 1.175494E-38

Maximum float positive value: 3.402823E+38

Precision value: 6
```

The void Type

The void type specifies that no value is available. It is used in three kinds of situations:



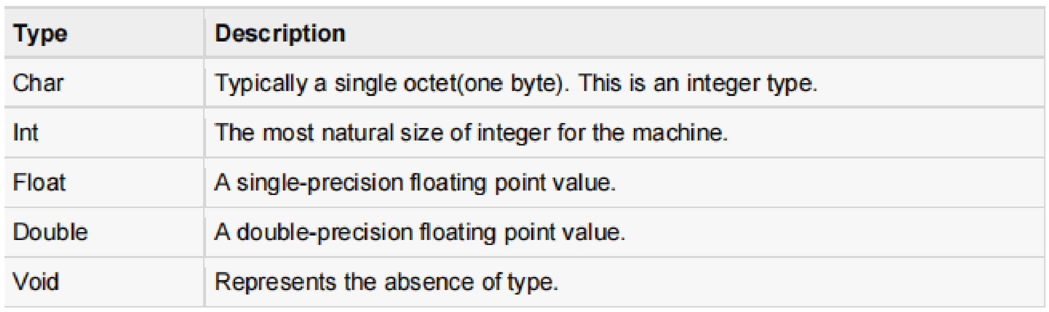
S.N.	Types and Description
1	**Function returns as void** There are various functions in C which do not return value or you can say they return void. A function with no return value has the return type as void. For example, **void exit (int status);**
2	**Function arguments as void** There are various functions in C which do not accept any parameter. A function with no parameter can accept as a void. For example, **int rand(void);**
3	**Pointers to void** A pointer of type void * represents the address of an object, but not its type. For example, a memory allocation function **void *malloc(size_t size);** returns a pointer to void which can be casted to any data type.

C VARIABLES

A variable is nothing but a name given to a storage area that our programs can manipulate. Each variable in C has a specific type, which determines the size and layout of the variable's memory; the range of values that can be stored within that memory; and the set of operations that can be applied to the variable.

The name of a variable can be composed of letters, digits, and the underscore character. It must begin with either a letter or an underscore. Upper and lowercase letters are distinct because C is case-sensitive. Based on the basic types explained in previous chapter, there will be the following basic variable types:

Type
Description



Type	Description
Char	Typically a single octet(one byte). This is an integer type.
Int	The most natural size of integer for the machine.
Float	A single-precision floating point value.
Double	A double-precision floating point value.
Void	Represents the absence of type.

C programming language also allows to define various other types of variables, which we will cover in subsequent chapters like Enumeration, Pointer, Array, Structure, Union, etc.

For this chapter, let us study only basic variable types.

Variable Definition in C:

A variable definition means to tell the compiler where and how much to create the storage for the variable. A variable definition specifies a data type and contains a list of one or more variables of that type as follows:

```
type variable_list;
```

Here, **type** must be a valid C data type including char, w_char, int, float, double, bool or any userdefined object, etc., and **variable_list** may consist of one or more identifier names separated by commas. Some valid declarations are shown here:

```
int    i, j, k;
char   c, ch;
float  f, salary;
double d;
```

The line **int i, j, k;** both declares and defines the variables i, j and k; which instructs the compiler to create variables named i, j and k of type int. Variables can be initialized (assigned an initial value) in their declaration. The initializer consists of an equal sign followed by a constant expression as follows:

```
type variable_name = value;
```

Some examples are:

```
extern int d = 3, f = 5;    // declaration of d and f.
int d = 3, f = 5;           // definition and initializing d and f.
byte z = 22;                // definition and initializes z.
char x = 'x';               // the variable x has the value 'x'.
```

For definition without an initializer: variables with static storage duration are implicitly initialized with NULL (all bytes have the value 0); the initial value of all other variables is undefined.

Variable Declaration in C:

A variable declaration provides assurance to the compiler that there is one variable existing with the given type and name so that compiler proceed for further compilation without needing complete detail about the variable. A variable declaration has its meaning at the time of compilation only, compiler needs actual variable declaration at the time of linking of the program. A variable declaration is useful when you are using multiple files and you define your variable in one of the files, which will be available at the time of linking of the program. You will use extern keyword to declare a variable at any place. Though you can declare a variable multiple times in your C program but it can be defined only once in a file, a function or a block of code.

Example

Try the following example, where variables have been declared at the top, but they have been defined and initialized inside the main function:

```
#include <stdio.h>

// Variable definition:
extern int a, b;
extern int c;
extern float f;

int main ()
{
// Variable definition:
int a, b;
int c;
float f;

// actual initialization
  a =10;
```

```
  b =20;
  c = a + b;

  printf("value of c : %d \n", c);

  f = 70.0/3.0;
  printf("value of f : %f \n", f);

  return 0;
}
```

When the above code is compiled and executed, it produces the following result:

```
value of c : 30

value of f : 23.333334
```

Same concept applies on function declaration where you provide a function name at the time of its declaration and its actual definition can be given anywhere else. For example:

```
// function declaration
int func();

int main()
{
   // function call
    int i = func();
}
// function definition
int func()
{
   return 0;
}
```

Lvalues and Rvalues in C

There are two kinds of expressions in C:

1.**lvalue:** An expression that is an **lvalue** may appear as either the left-hand or right-hand side of an assignment.

2.**rvalue:** An expression that is an **rvalue** may appear on the right- but not left-hand side of an assignment.

Variables are lvalues and so may appear on the left-hand side of an assignment. Numeric literals are rvalues and so may not be assigned and cannot appear on the left-hand side. Following is a valid statement:

```
int g = 20;
```

But following is not a valid statement and would generate compile-time error:

```
10 = 20;
```

CONSTANTS

If you don't want others (or yourself) to change existing variable values, you can use the const keyword. This will declare the variable as "constant", which means unchangeable and read-only:

```
const int myNum = 15;  // myNum will always be 15
myNum = 10;  // error: assignment of read-only variable 'myNum'
```

C OPERATORS

C divides the operators into the following groups:

1. Arithmetic operators
2. Assignment operators
3. Comparison operators
4. Logical operators
5. Bitwise operators

Arithmetic Operators

Arithmetic operators are used to perform common mathematical operations.

+ , - , * , / , % , ++ , --

Assignment Operators

Assignment operators are used to assign values to variables.

== , += , -= , *= , /= , %= , &= , |= , ^= , >>= , <<=

Comparison Operators

Comparison operators are used to compare two values (or variables). This is important in programming, because it helps us to find answers and make decisions.

The return value of a comparison is either 1 or 0, which means **true** (1) or **false** (0). These values are known as **Boolean values.**

== , != , > , < , >= , <=

Logical Operators

You can also test for true or false values with logical operators.

Logical operators are used to determine the logic between variables or values

&& , || , !

Sizeof Operator

The memory size (in bytes) of a data type or a variable can be found with the sizeof operator

DECISION MAKING IN C

In decision making we use certain statements to control the flow of the action of code

1. if statement
2. else statement
3. elseif statements
4. Switch

if statement

An **if** statement consists of a boolean expression followed by one or more statements.

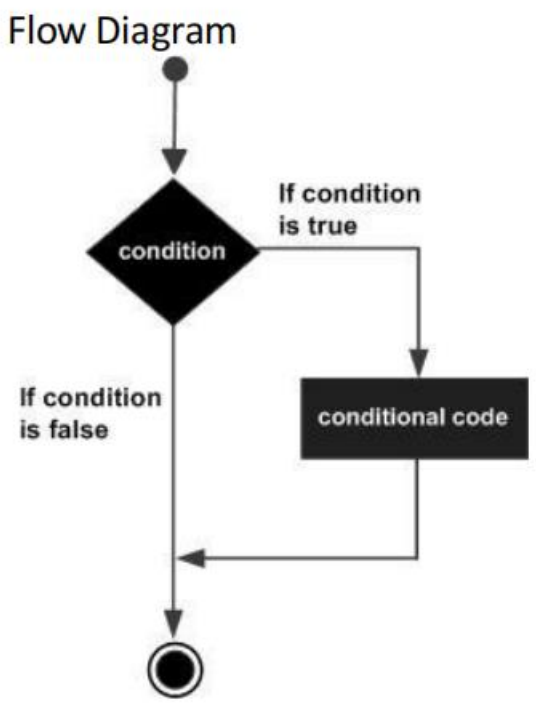
Syntax

The syntax of an if statement in C programming language is:

```
if(boolean_expression)
{
   /* statement(s) will execute if the boolean expression is true */
}
```

If the **boolean** expression evaluates to **true,** then the block of code inside the **if** statement will be executed. If **boolean** expression evaluates to **false,** then the first set of code after the end of the **if** statement (after the closing curly brace) will be executed.

C programming language assumes any **non-zero** and **non-null** values as **true** and if it is either **zero** or **null** then it is assumed as **false** value.



```
#include <stdio.h>

int main ()
{
   /* local variable definition */
   int a = 10;

   /* check the boolean condition using if statement */
   if( a < 20 )
   {
       /* if condition is true then print the following */
       printf("a is less than 20\n" );
   }
   printf("value of a is : %d\n", a);

   return 0;
}
```

When the above code is compiled and executed, it produces the following result

```
a is less than 20;

value of a is : 10
```

if...else statement

An **if** statement can be followed by an optional **else** statement, which executes when the boolean expression is **false**.

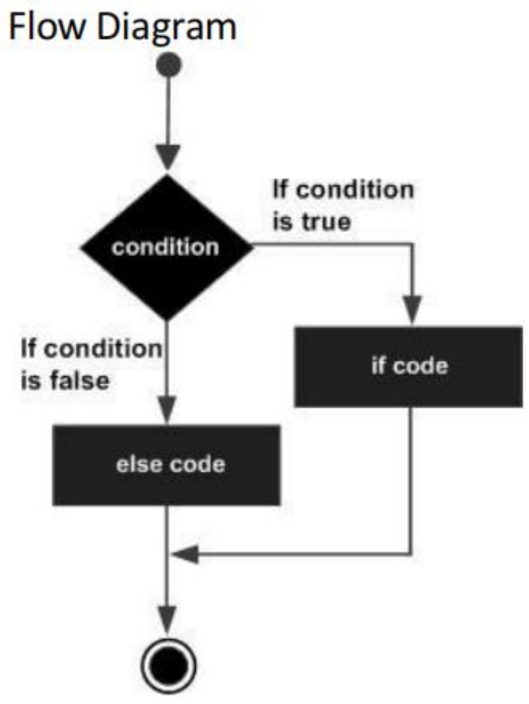
Syntax

The syntax of an **if...else** statement in C programming language is:

```
if(boolean_expression)
{
   /* statement(s) will execute if the boolean expression is true */
}
else
{
  /* statement(s) will execute if the boolean expression is false */
}
```

If the **boolean** expression evaluates to **true,** then the if block of code will be executed, otherwise **else** block of code will be executed.

C programming language assumes any **non-zero** and **non-null** values as **true** and if it is either **zero** or **null** then it is assumed as **false** value.



```
#include <stdio.h>

int main ()
{
   /* local variable definition */
   int a = 100;

   /* check the boolean condition */
   if( a < 20 )
   {
       /* if condition is true then print the following */
       printf("a is less than 20\n" );
   }
   else
   {
       /* if condition is false then print the following */
       printf("a is not less than 20\n" );
   }
   printf("value of a is : %d\n", a);

   return 0;
}
```

When the above code is compiled and executed, it produces the following result:

```
a is not less than 20;

value of a is : 100
```

Switch statement

A **switch** statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each switch case.

Syntax

The syntax for a **switch** statement in C programming language is as follows:

```
switch(expression){
    case constant-expression  :
       statement(s);
       break; /* optional */
    case constant-expression  :
       statement(s);
       break; /* optional */

    /* you can have any number of case statements */
    default : /* Optional */
       statement(s);
}
```

The following rules apply to a **switch** statement:

1. The **expression** used in a **switch** statement must have an integral or enumerated type, or be of a class type in which the class has a single conversion function to an integral or enumerated type.

□

2. You can have any number of case statements within a switch. Each case is followed by the value to be compared to and a colon.

□

3. The **constant-expression** for a case must be the same data type as the variable in the switch, and it must be a constant or a literal.

□

4. When the variable being switched on is equal to a case, the statements following that case will execute until a **break** statement is reached.

□

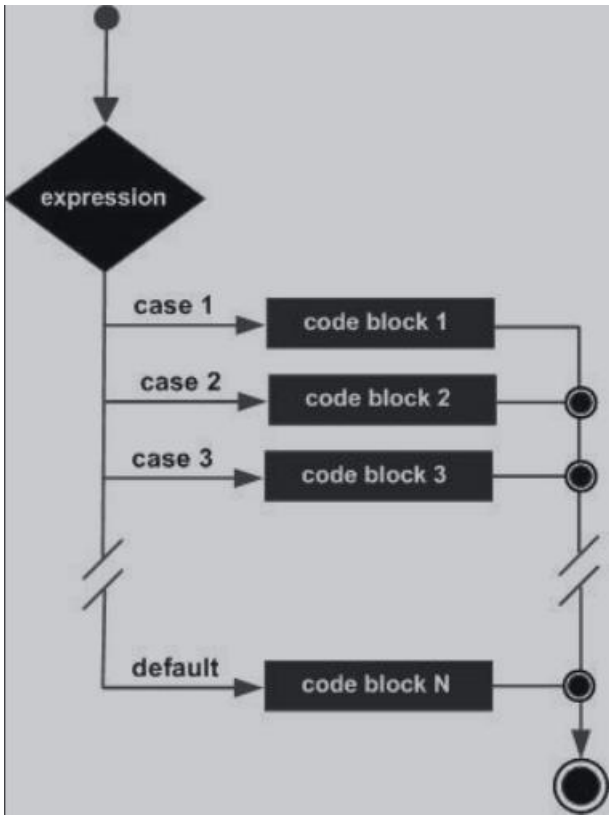
5. When a **break** statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.

□

6. Not every case needs to contain a **break**. If no **break** appears, the flow of control will fall through to subsequent cases until a break is reached.

□

7. A **switch** statement can have an optional **default** case, which must appear at the end of the switch. The default case can be used for performing a task when none of the cases is true. No **break** is needed in the default case.



```
#include <stdio.h>

int main ()
{
   /* local variable definition */
   char grade = 'B';

   switch(grade)
   {
   case 'A' :
      printf("Excellent!\n" );
      break;
   case 'B' :
   case 'C' :
      printf("Well done\n" );
      break;
   case 'D' :
      printf("You passed\n" );
      break;
   case 'F' :
      printf("Better try again\n" );
      break;
   default :
      printf("Invalid grade\n" );
   }
   printf("Your grade is  %c\n", grade );

   return 0;
}
```

When the above code is compiled and executed, it produces the following result:

HEADER FILES USED

1. stdio.h: The C standard library is a collection of functions, which are declared in header files, and stdio.h is one of them. The name stands for "Standard Input Output", so in that file you can find all the declarations of functions that deal with input, output and files. You can find a list of the header files included in the C standard library here. A library is a compiled binary file (or in general, a collection of binary files), which can be linked when compiling a program to take advantage of the functions made available by the library (i.e. exported). Header files are then used to identify the names and the signatures of those functions, so that the compiler knows how to call them

2. string.h: string.h is the header in the C standard library for the C programming language which contains macro definitions, constants and declarations of functions and types used not only for string handling but also various memory handling functions; the name is thus something of a misnomer.

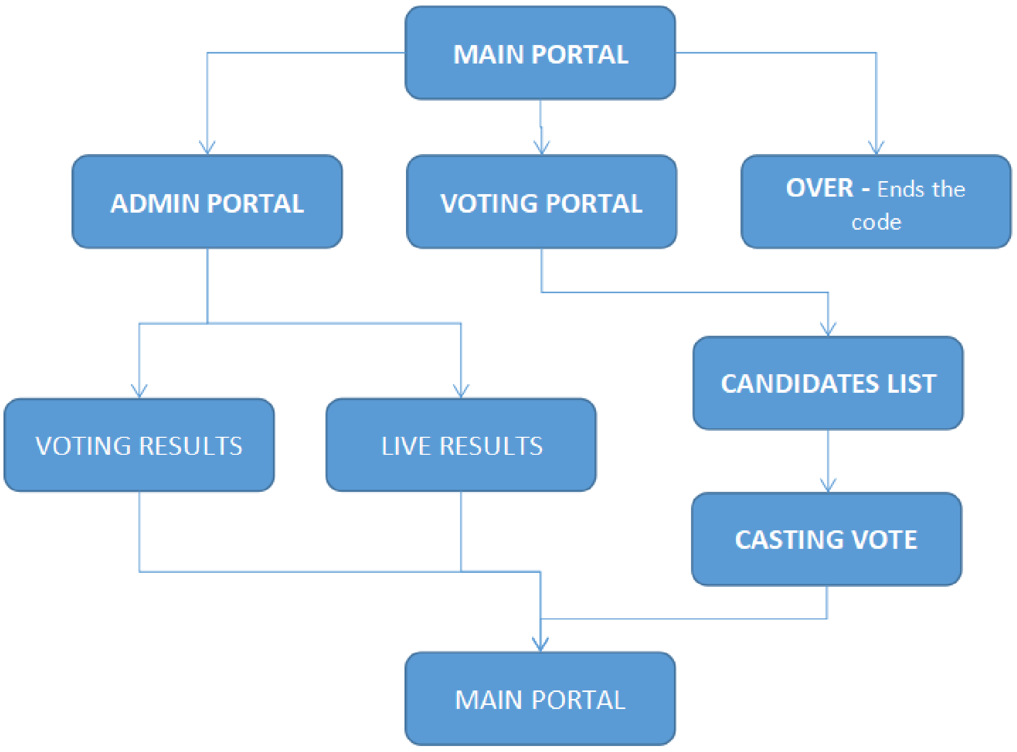
WORKING

When we run the code it opens the main portal where we get 3 options **1.admin portal , 2.voting portal , 3.over**. When we enter the number 1 It take us to the admin portal where it asks the user name and password , and it grant access when we enter the correct details , where it shows 2 options 1.voting results , 2.live voting . in voting results it shows who has won the election , and in live voting it shows who is on the leading .

In admin portal the 2nd option is voting portal where it displays the voter names . we have to select our name and enter our credentials , it will give access when the given credentials are correct. The voter can choose the candidate and cast the vote . and it return to the man portal. And the last option is over this ends the voting.

This is a effective way to cast votes and viewing the results.

DIAGRAMATIC REPESENTATION OF OUTPUT



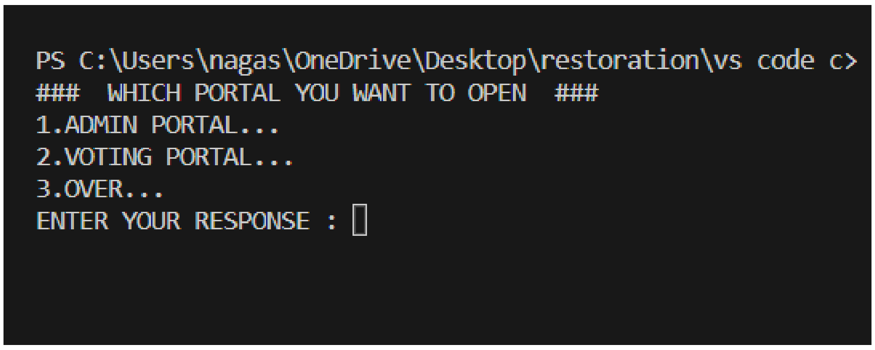
CODE

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void main_portal();
void candidate_list();
void error();
void error1();
void admin_pass();
void voter_names();
void voting_results();
void admin_portal();
void live_voting();
void votes_after_results();

char a_[] = " 1" , b_[] = " 2" , c_[] = " 3" , d_[] = " 4" , e_[] = " 5" ;
char f_[] = " 6" , g_[] = " 7" , h_[] = " 8" , i_[] = " 9" , j_[] = " 10" ;

char voter_check[20] = {"$"};

int CANDIDATE1_recieved=0 , CANDIDATE2_recieved=0 , NOTA_recieved=0;
#define CANDIDATE1 "PREETHAM"
#define CANDIDATE2 "PREM"
#define NOTA "NOTA"

// live voting
void live_voting(){
   printf("\n### live votes ###\n");
   printf("\n%s WITH %d VOTES" , CANDIDATE1, CANDIDATE1_recieved);
   printf("\n%s WITH %d VOTES" , CANDIDATE2, CANDIDATE2_recieved);
   printf("\nNO. OF VOTES SPOILED %d \n\n" , NOTA_recieved );
   admin_portal();
}

// VOTES RECIEVED BY ANOTHER CANDIDATES
void votes_after_results(){
   printf("\n### votes ###\n");
   printf("\n%s WITH %d VOTES" , CANDIDATE1, CANDIDATE1_recieved);
   printf("\n%s WITH %d VOTES" , CANDIDATE2, CANDIDATE2_recieved);
   printf("\nNO. OF VOTES SPOILED : %d\n\n" , NOTA_recieved );
}

//ADMIN PORTAL
void admin_portal(){
   int option;
   printf("\n1.VOTING RESULTS\n");
```

```
    printf("2.LIVE RESULTS\n");
    printf("3.BACK");
    printf("ENTER YOUR RESPONSE : ");
    scanf("%d" , &option);
    switch(option){
        case 1 : voting_results(); break;
        case 2 : live_voting(); break;
        case 3 : main_portal(); break;
        default : printf("PLEASE ENTER A VALID NUMBER");
    }

}

//VOTING RESULTS
void voting_results(){
    printf ("\n### VOTING RESULTS ###\n\n");
    if(CANDIDATE1_recieved > CANDIDATE2_recieved &&
CANDIDATE1_recieved > NOTA_recieved){
        printf("\033[92m");
        printf("%s HAS WON THE ELECTIONS WITH %d VOTES\n\n",
CANDIDATE1,CANDIDATE1_recieved);
        printf("\033[0m");
    }
    else if( NOTA_recieved == CANDIDATE2_recieved &&
CANDIDATE1_recieved == NOTA_recieved){
        printf("\033[91m");
        printf("NO. OF VOTES RECIEVED BY BOTH THE COMPETITORS AND
NO. OF VOTES SPOILED ARE EQUAL \n");
        printf("IT'S TIME FOR THE RE-ELECTION\n\n");
        printf("\033[0m");
    }
    else if(NOTA_recieved > CANDIDATE2_recieved && CANDIDATE1_recieved
< NOTA_recieved){
        printf("\033[91m");
        printf("MOST OF THE VOTES WERE SPOILED\n");
        printf("IT'S TIME FOR THE RE-ELECTION\n\n");
        printf("\033[0m");
    }
    else if(CANDIDATE1_recieved > NOTA_recieved && CANDIDATE2_recieved >
NOTA_recieved && CANDIDATE1_recieved == CANDIDATE2_recieved ){
        printf("\033[91m");
        printf("BOTH THE COMPETITORS HAS RECIEVED SAME NUMBER OF
VOTES\n");
        printf("IT'S TIME FOR THE RE-ELECTION\n\n");
        printf("\033[0m");
    }
    else if(CANDIDATE1_recieved < CANDIDATE2_recieved &&
CANDIDATE2_recieved > NOTA_recieved){
        printf("\033[92m");
        printf("%s HAS WON THE ELECTIONS WITH %d VOTES\n\n",
CANDIDATE2,CANDIDATE2_recieved);
        printf("\033[0m");
    }

    votes_after_results();
}

//error program
void error(){
    int else9;
    printf ("\nSESSION TERMINATED\n\n");
    printf ("ENTER 1 TO RETURN TO MAIN PORTAL\n");
    printf ("ENTER 2 TO RETURN TO VOTER PORTAL\n");
    // printf ("ENTER 0 TO STOP\n");
    printf ("ENTER YOUR RESPONSE HERE : ");
    scanf("%d", &else9);
    switch(else9){
        case 1 : main_portal();break;
        case 2 : voter_names();break;
        // case 3 : printf("THANKYOU\n");break;
        return ; printf("PLEASE ENTER A VALIED NUMBER\n"); void error();
    }
}

void error1(){
    int else9;
    printf ("\nSESSION TERMINATED\n\n");
    printf ("ENTER 1 TO RETRY\n");
    printf ("ENTER 2 TO RETURN TO MAIN PORTAL\n");
    // printf ("ENTER 0 TO STOP\n");
    printf ("ENTER YOUR RESPONSE HERE : ");
    scanf("%d", &else9);
    switch(else9){
        case 1 : admin_pass();break;
        case 2 : main_portal();break;
        // case 3 : printf("THANKYOU\n");break;
        return ; printf("PLEASE ENTER A VALIED NUMBER\n"); void error1();
    }
}


//MAIN PORTAL

void admin_pass(){
    char username[30] = {'L','A','K','S','H','M','A','N'};
    char user_check[30];
    printf("\nUSERNAME : ");
    scanf ("%s" , &user_check);
    if(strcmp(username,user_check)==0){
        char password[30] = {'L','A','K','S','H','M','A','N','1','2','3'};
        char pass_check[30];
        printf ("ENTER THE PASSWORD : ");
        scanf ("%s" , &pass_check);
        if (strcmp(password,pass_check) == 0){
            admin_portal();
        }
        else {
            error1();
        }
    }

    else{
        error1();
    }

}

//candidate list
void candidate_list() {
    int a;
    printf("\n\n ### Please choose your Candidate ####\n\n");
    printf("1. %s\n", CANDIDATE1);
    printf("2. %s\n", CANDIDATE2);
    printf("3. %s\n", NOTA);
    printf("ENTER THE NUMBER BASED ON YOUR CHOICE (1-3) :");
    scanf("%d" , &a);
    switch(a){
        case 1 : CANDIDATE1_recieved++ , printf("\n### THANKYOU FOR
VOTING ###\n\n"); main_portal(); break;
        case 2 : CANDIDATE2_recieved++ , printf("\n### THANKYOU FOR
VOTING ###\n\n"); main_portal(); break;
        case 3 : NOTA_recieved++ ,  printf("\n### THANKYOU FOR VOTING
###\n\n"); main_portal(); break;
        default : printf("PLEASE ENTER A VALID NUMBER\n\n"); candidate_list();

    }

}


//VOTERS
//VOTER 1
void PREETHAM_V1(){
    char VOTER1_USERNAME[30] = {'2','2','1','8','0','1','3','5','0','0','0','7'};
    char VOTER1_USERNAME_TALLY[30];
    printf("\nUSERNAME : ");
    scanf("%s" , &VOTER1_USERNAME_TALLY);
    if (strcmp(VOTER1_USERNAME,VOTER1_USERNAME_TALLY)==0){
        char VOTER1_PASSWORD[20] = {'P','R','E','E','T','H','A','M','@'};
        char VOTER1_PASSWORD_TALLY[20];
        printf ("ENTER THE PASSWORD : ");
        scanf("%s" , &VOTER1_PASSWORD_TALLY);
        if (strcmp(VOTER1_PASSWORD,VOTER1_PASSWORD_TALLY) == 0){
            char VOTER1_FP[5] = {'@','1'};
            char VOTER1_FP_TALLY[5];
            printf("\nPLACE YOUR FINGRER ON THE CENSOR : ");
            scanf("%s" , &VOTER1_FP_TALLY);
            if(strcmp (VOTER1_FP,VOTER1_FP_TALLY)==0){
                printf("ACCESS GRANTED\n");
                strcat(voter_check,a_);printf("%s" ,voter_check);
                candidate_list();
            }
            else {
                error();
            }
        }
        else{
            error();
        }
    }

    else{
        error();
    }
}


//VOTER 2
void PREM_V2(){
    char VOTER2_USERNAME[30] = {'2','2','1','8','0','1','3','9','0','0','0','8'};
    char VOTER2_USERNAME_TALLY[30];
    printf("\nUSERNAME : ");
    scanf("%s" , &VOTER2_USERNAME_TALLY);
    if (strcmp(VOTER2_USERNAME,VOTER2_USERNAME_TALLY)==0){
        char VOTER2_PASSWORD[20] = {'P','R','E','M','@'};
        char VOTER2_PASSWORD_TALLY[20];
        printf ("ENTER THE PASSWORD : ");
        scanf("%s" , &VOTER2_PASSWORD_TALLY);
        if (strcmp(VOTER2_PASSWORD,VOTER2_PASSWORD_TALLY) == 0){
            char VOTER1_FP[5] = {'@','2'};
            char VOTER1_FP_TALLY[5];
            printf("\nPLACE YOUR FINGRER ON THE CENSOR : ");
            scanf("%s" , &VOTER1_FP_TALLY);
            if(strcmp (VOTER1_FP,VOTER1_FP_TALLY)==0){
                printf("ACCESS GRANTED\n");
                strcat(voter_check,b_);printf("%s" ,voter_check);
                candidate_list();
            }
            else {
                error();
            }
        }
        else{
            error();
        }
    }
    else{
        error();
    }
}


//VOTER 3
void BHASKAR_V3(){
    char VOTER3_USERNAME[30] = {'2','2','1','8','0','1','3','5','0','0','1','9'};
    char VOTER3_USERNAME_TALLY[30];
    printf("\nUSERNAME : ");
    scanf("%s" , &VOTER3_USERNAME_TALLY);
    if (strcmp(VOTER3_USERNAME,VOTER3_USERNAME_TALLY)==0){
        char VOTER3_PASSWORD[20] = {'B','H','A','S','K','A','R','@'};
        char VOTER3_PASSWORD_TALLY[20];
        printf ("ENTER THE PASSWORD : ");
        scanf("%s" , &VOTER3_PASSWORD_TALLY);
        if (strcmp(VOTER3_PASSWORD,VOTER3_PASSWORD_TALLY) == 0){
            char VOTER1_FP[5] = {'@','3'};
            char VOTER1_FP_TALLY[5];
            printf("\nPLACE YOUR FINGRER ON THE CENSOR : ");
            scanf("%s" , &VOTER1_FP_TALLY);
            if(strcmp (VOTER1_FP,VOTER1_FP_TALLY)==0){
                printf("ACCESS GRANTED\n");
                strcat(voter_check,c_);printf("%s" ,voter_check);
                candidate_list();
            }
            else {
                error();
            }
        }
        else{
            error();
        }
    }
    else{
        error();
    }
}


//VOTER 4
void ROHITH_V4(){
    char VOTER4_USERNAME[30] = {'2','2','1','8','0','1','3','5','0','0','1','7'};
    char VOTER4_USERNAME_TALLY[30];
    printf("\nUSERNAME : ");
    scanf("%s" , &VOTER4_USERNAME_TALLY);
    if (strcmp(VOTER4_USERNAME,VOTER4_USERNAME_TALLY)==0){
        char VOTER4_PASSWORD[20] = {'R','O','H','I','T','H','@'};
        char VOTER4_PASSWORD_TALLY[20];
        printf ("ENTER THE PASSWORD : ");
        scanf("%s" , &VOTER4_PASSWORD_TALLY);
        if (strcmp(VOTER4_PASSWORD,VOTER4_PASSWORD_TALLY) == 0){
            char VOTER1_FP[5] = {'@','4'};
            char VOTER1_FP_TALLY[5];
            printf("\nPLACE YOUR FINGRER ON THE CENSOR : ");
            scanf("%s" , &VOTER1_FP_TALLY);
            if(strcmp (VOTER1_FP,VOTER1_FP_TALLY)==0){
                printf("ACCESS GRANTED\n");
                strcat(voter_check,d_);printf("%s" ,voter_check);
                candidate_list();
            }
            else {
                error();
            }
        }
        else{
            error();
        }
    }

    else{
        error();
    }
}


//VOTER 5
void SAI_KIRAN_V5(){
    char VOTER5_USERNAME[30] = {'2','2','1','8','0','1','3','5','0','0','0','2'};
    char VOTER5_USERNAME_TALLY[30];
    printf("\nUSERNAME : ");
    scanf("%s" , &VOTER5_USERNAME_TALLY);
    if (strcmp(VOTER5_USERNAME,VOTER5_USERNAME_TALLY)==0){
        char VOTER5_PASSWORD[20] = {'S','A','I','K','I','R','A','N','@'};
        char VOTER5_PASSWORD_TALLY[20];
        printf ("ENTER THE PASSWORD : ");
        scanf("%s" , &VOTER5_PASSWORD_TALLY);
        if (strcmp(VOTER5_PASSWORD,VOTER5_PASSWORD_TALLY) == 0){
            char VOTER1_FP[5] = {'@','5'};
            char VOTER1_FP_TALLY[5];
            printf("\nPLACE YOUR FINGRER ON THE CENSOR : ");
            scanf("%s" , &VOTER1_FP_TALLY);
            if(strcmp (VOTER1_FP,VOTER1_FP_TALLY)==0){
                printf("ACCESS GRANTED\n");
                strcat(voter_check,e_);printf("%s" ,voter_check);
                candidate_list();
            }
            else {
                error();
            }
        }
        else{
            error();
        }
    }

    else{
        error();
    }
}


//VOTER 6
void SANDESH_V6(){
    char VOTER6_USERNAME[30] = {'2','2','1','8','0','1','3','9','0','0','0','3'};
    char VOTER6_USERNAME_TALLY[30];
    printf("\nUSERNAME : ");
    scanf("%s" , &VOTER6_USERNAME_TALLY);
    if (strcmp(VOTER6_USERNAME,VOTER6_USERNAME_TALLY)==0){
        char VOTER6_PASSWORD[20] = {'S','A','N','D','E','S','H','@'};
        char VOTER6_PASSWORD_TALLY[20];
        printf ("ENTER THE PASSWORD : ");
        scanf("%s" , &VOTER6_PASSWORD_TALLY);
        if (strcmp(VOTER6_PASSWORD,VOTER6_PASSWORD_TALLY) == 0){
            char VOTER1_FP[5] = {'@','6'};
            char VOTER1_FP_TALLY[5];
            printf("\nPLACE YOUR FINGRER ON THE CENSOR : ");
            scanf("%s" , &VOTER1_FP_TALLY);
            if(strcmp (VOTER1_FP,VOTER1_FP_TALLY)==0){
                printf("ACCESS GRANTED\n");
                strcat(voter_check,f_);printf("%s" ,voter_check);
                candidate_list();
            }
            else {
                error();
            }
        }
        else{
            error();
        }
    }
    else{
        error();
    }
}


//VOTER 7
void CHARAN_V7(){
    char VOTER7_USERNAME[30] = {'2','2','1','8','0','1','3','5','0','0','1','1'};
    char VOTER7_USERNAME_TALLY[30];
    printf("\nUSERNAME : ");
    scanf("%s" , &VOTER7_USERNAME_TALLY);
    if (strcmp(VOTER7_USERNAME,VOTER7_USERNAME_TALLY)==0){
        char VOTER7_PASSWORD[20] = {'C','H','A','R','A','N','@'};
        char VOTER7_PASSWORD_TALLY[20];
        printf ("ENTER THE PASSWORD : ");
        scanf("%s" , &VOTER7_PASSWORD_TALLY);
        if (strcmp(VOTER7_PASSWORD,VOTER7_PASSWORD_TALLY) == 0){
            char VOTER1_FP[5] = {'@','7'};
            char VOTER1_FP_TALLY[5];
            printf("\nPLACE YOUR FINGRER ON THE CENSOR : ");
            scanf("%s" , &VOTER1_FP_TALLY);
            if(strcmp (VOTER1_FP,VOTER1_FP_TALLY)==0){
                printf("ACCESS GRANTED\n");
                strcat(voter_check,g_);printf("%s" ,voter_check);
                candidate_list();
            }
            else {
                error();
            }
        }
        else{
            error();
        }
    }

    else{
        error();
    }
}


//VOTER 8
void SRIYA_V8(){
    char VOTER8_USERNAME[30] = {'2','2','1','8','0','1','3','6','0','0','1','3'};
    char VOTER8_USERNAME_TALLY[30];
    printf("\nUSERNAME : ");
    scanf("%s" , &VOTER8_USERNAME_TALLY);
    if (strcmp(VOTER8_USERNAME,VOTER8_USERNAME_TALLY)==0){
        char VOTER8_PASSWORD[20] = {'S','R','I','Y','A','@'};
        char VOTER8_PASSWORD_TALLY[20];
        printf ("ENTER THE PASSWORD : ");
        scanf("%s" , &VOTER8_PASSWORD_TALLY);
        if (strcmp(VOTER8_PASSWORD,VOTER8_PASSWORD_TALLY) == 0){
            char VOTER1_FP[5] = {'@','8'};
            char VOTER1_FP_TALLY[5];
            printf("\nPLACE YOUR FINGRER ON THE CENSOR : ");
            scanf("%s" , &VOTER1_FP_TALLY);
            if(strcmp (VOTER1_FP,VOTER1_FP_TALLY)==0){
                printf("ACCESS GRANTED\n");
                strcat(voter_check,h_);printf("%s" ,voter_check);
                candidate_list();
            }
            else {
                error();
            }
        }
        else{
            error();
        }
    }

    else{
        error();
    }
}


//VOTER 9
void MADHURI_V9(){
    char VOTER9_USERNAME[30] = {'2','2','1','8','0','1','3','5','0','0','1','0'};
    char VOTER9_USERNAME_TALLY[30];
    printf("\nUSERNAME : ");
    scanf("%s" , &VOTER9_USERNAME_TALLY);
    if (strcmp(VOTER9_USERNAME,VOTER9_USERNAME_TALLY)==0){
        char VOTER9_PASSWORD[20] = {'M','A','D','U','R','I','@'};
        char VOTER9_PASSWORD_TALLY[20];
        printf ("ENTER THE PASSWORD : ");
        scanf("%s" , &VOTER9_PASSWORD_TALLY);
        if (strcmp(VOTER9_PASSWORD,VOTER9_PASSWORD_TALLY) == 0){
            char VOTER1_FP[5] = {'@','9'};
            char VOTER1_FP_TALLY[5];
            printf("\nPLACE YOUR FINGRER ON THE CENSOR : ");
            scanf("%s" , &VOTER1_FP_TALLY);
            if(strcmp (VOTER1_FP,VOTER1_FP_TALLY)==0){
                printf("ACCESS GRANTED\n");
                strcat(voter_check,i_);printf("%s" ,voter_check);
                candidate_list();
            }
            else {
                error();
            }
        }
        else{
            error();
        }
    }
    else{
        error();
    }
}


//VOTER 10
void NIHARIKA_V10(){
    char VOTER10_USERNAME[30] = {'2','2','1','8','0','1','3','5','0','0','0','8'};
    char VOTER10_USERNAME_TALLY[30];
    printf("\nUSERNAME : ");
    scanf("%s" , &VOTER10_USERNAME_TALLY);
    if (strcmp(VOTER10_USERNAME,VOTER10_USERNAME_TALLY)==0){
        char VOTER10_PASSWORD[20] = {'N','I','H','A','R','I','K','A','@'};
        char VOTER10_PASSWORD_TALLY[20];
        printf ("ENTER THE PASSWORD : ");
        scanf("%s" , &VOTER10_PASSWORD_TALLY);
        if (strcmp(VOTER10_PASSWORD,VOTER10_PASSWORD_TALLY) == 0){
            char VOTER1_FP[5] = {'@','1','0'};
            char VOTER1_FP_TALLY[5];
            printf("\nPLACE YOUR FINGRER ON THE CENSOR : ");
            scanf("%s" , &VOTER1_FP_TALLY);
            if(strcmp (VOTER1_FP,VOTER1_FP_TALLY)==0){
                printf("ACCESS GRANTED\n");
                strcat(voter_check,j_);printf("%s" ,voter_check);
                candidate_list();
            }
            else {
                error();
            }
        }
        else{
            error();
        }
    }

    else{
        error();
    }
}


//voter names
```

```
void voter_names(){
    #define voter_1 "PREETHAM"
    #define voter_2 "PREM"
    #define voter_3 "BHASKAR"
    #define voter_4 "ROHITH"
    #define voter_5 "SAI KIRAN"
    #define voter_6 "SANDESH"
    #define voter_7 "CHARAN"
    #define voter_8 "SRIYA"
    #define voter_9 "MADHURI"
    #define voter_10 "NIHARIKA"

    int vote_num;
    char ch[3];
    printf("\n\n ### SELECT YOUR NAME ###\n");
    printf("1. %s\n" , voter_1);
    printf("2. %s\n" , voter_2);
    printf("3. %s\n" , voter_3);
    printf("4. %s\n" , voter_4);
    printf("5. %s\n" , voter_5);
    printf("6. %s\n" , voter_6);
    printf("7. %s\n" , voter_7);
    printf("8. %s\n" , voter_8);
    printf("9. %s\n" , voter_9);
    printf("10. %s\n" , voter_10);
    printf("ENTER YOUR RESPONSE : ");
    scanf("%s" , &ch);


    char *ptr ;
    ptr = strstr(voter_check,ch);
    vote_num = atoi(ch);

    if (ptr != NULL){
        printf("\nU HAVE ALREADY VOTED\n\n");main_portal();
    }
    else{
        strcat(voter_check,ch);
        switch(vote_num){
            case 1 : PREETHAM_V1(); break;
            case 2 : PREM_V2(); break;
            case 3 : BHASKAR_V3(); break;
            case 4 : ROHITH_V4(); break;
            case 5 : SAI_KIRAN_V5(); break;
            case 6 : SANDESH_V6(); break;
            case 7 : CHARAN_V7(); break;
            case 8 : SRIYA_V8(); break;
            case 9 : MADHURI_V9(); break;
            case 10 : NIHARIKA_V10(); break;
            default : printf("\nPLEASE GIVE THE CORRECT 
RESPONCE");voter_names();
        }
    }
}

//MAIN PORTAL
void main_portal() {
    int select;
    printf ("###  WHICH PORTAL YOU WANT TO OPEN  ###\n");
    printf ("1.ADMIN PORTAL...\n");
    printf ("2.VOTING PORTAL...\n");
    printf ("3.OVER...\n");
    printf ("ENTER YOUR RESPONSE : ");
    scanf ("%d" , &select);

    switch (select){
        case 1 : admin_pass();
        break;
        case 2 : voter_names();
        break;
        case 3 : printf ("### THANKYOU ###");
        break;
        default : printf("ENTER THE CORRECT OPTION");
    }

}

int main(){
    main_portal();
}
```

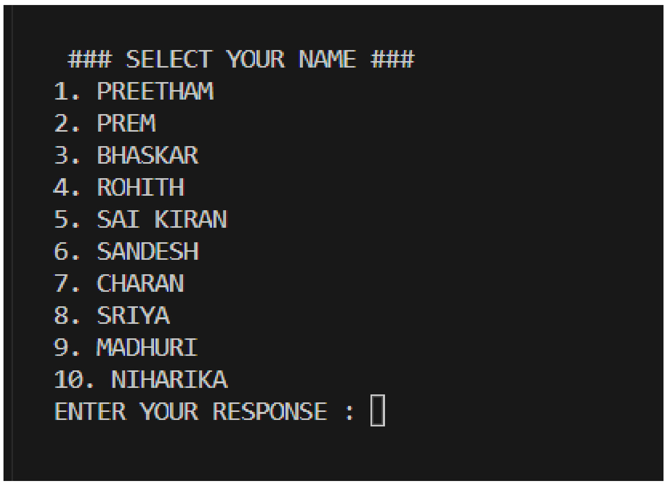
OUTPUT SCREENS



```
PS C:\Users\nagas\OneDrive\Desktop\restoration\vs code c>
###  WHICH PORTAL YOU WANT TO OPEN  ###
1.ADMIN PORTAL...
2.VOTING PORTAL...
3.OVER...
ENTER YOUR RESPONSE : ▯
```

Fig.1: This picture shows the main portal of the EVM

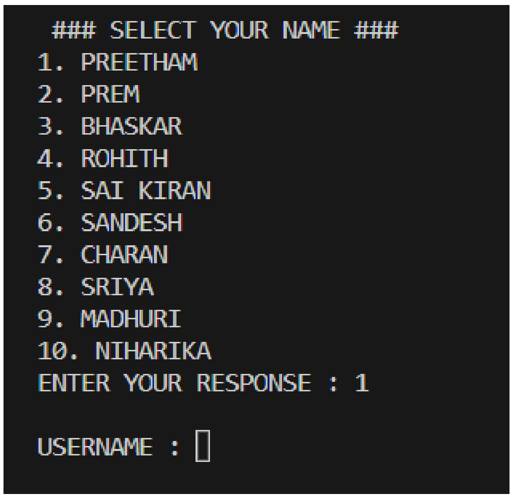
ENTERING TO THE VOTING PORTAL



```
 ### SELECT YOUR NAME ###
1. PREETHAM
2. PREM
3. BHASKAR
4. ROHITH
5. SAI KIRAN
6. SANDESH
7. CHARAN
8. SRIYA
9. MADHURI
10. NIHARIKA
ENTER YOUR RESPONSE : ▯
```

Fig.2: Here a list of voter's names has been displayed, we have to select your name and enter the number

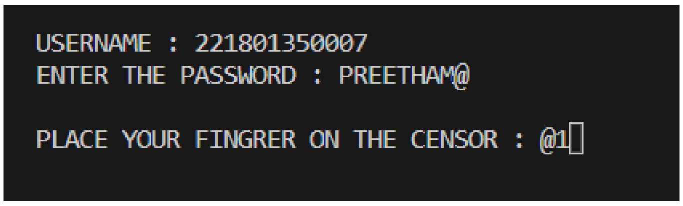
SELECTING A RANDOM VOTER



```
 ### SELECT YOUR NAME ###
1. PREETHAM
2. PREM
3. BHASKAR
4. ROHITH
5. SAI KIRAN
6. SANDESH
7. CHARAN
8. SRIYA
9. MADHURI
10. NIHARIKA
ENTER YOUR RESPONSE : 1

USERNAME : ▯
```

Fig.3: In this picture It displays the trial of the random voter

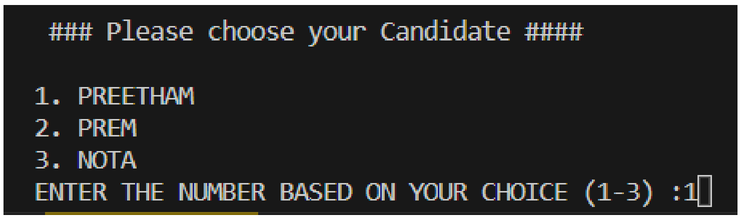
ENTERING THE VOTER CREDENTIALS



```
USERNAME : 221801350007
ENTER THE PASSWORD : PREETHAM@

PLACE YOUR FINGRER ON THE CENSOR : @1
```

Fig 4: here we have to enter the credentials

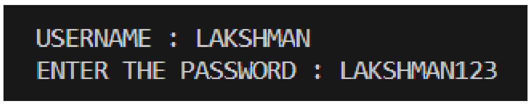
SELECTING THE DESIRED CANDIDATE



```
 ### Please choose your Candidate ####

1. PREETHAM
2. PREM
3. NOTA
ENTER THE NUMBER BASED ON YOUR CHOICE (1-3) :1
```

Fig.5: here it will display the candidates name, after selecting the desired candidate it will return to the main portal.

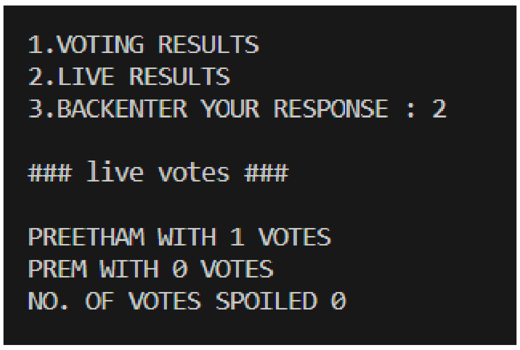
ENTERING THE ADMIN PORTAL CREDENTIALS



```
USERNAME : LAKSHMAN
ENTER THE PASSWORD : LAKSHMAN123
```

Fig 6: this picture asks the username and password to get access to the admin portal.

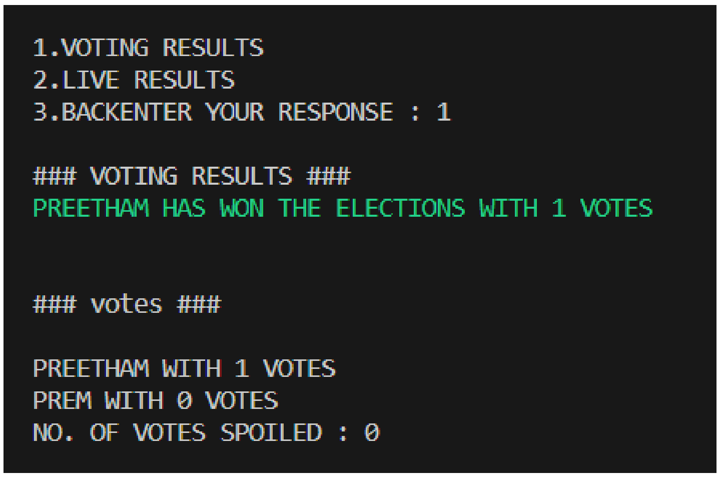
SELECTING THE LIVE RESULTS



```
1.VOTING RESULTS
2.LIVE RESULTS
3.BACKENTER YOUR RESPONSE : 2

### live votes ###

PREETHAM WITH 1 VOTES
PREM WITH 0 VOTES
NO. OF VOTES SPOILED 0
```

Fig 7: displays the number of votes achieved by the candidates

SELECTING THE VOTING RESULTS



```
1.VOTING RESULTS
2.LIVE RESULTS
3.BACKENTER YOUR RESPONSE : 1

### VOTING RESULTS ###
PREETHAM HAS WON THE ELECTIONS WITH 1 VOTES


### votes ###

PREETHAM WITH 1 VOTES
PREM WITH 0 VOTES
NO. OF VOTES SPOILED : 0
```

Fig 8: After selecting the voting results it will display the winner and the votes he has achieved.

CONCLUSION

We have successfully created a system that can accept votes from users and store them securely in a database.Electronic voting system is an effort in the direction to use computer for the purpose of voting system. It is both user friendly as well as time saving. Emphasis has been given in this application to replicate all the process required in a traditional system.

EVM system is self-sufficient with all the information required about the voters, candidate and the voters. Once the information is fed to the system, it can identify individual voters, their votes and then can press any key to return such as counting the votes has been simplified in this system.
The basic function of the polling has been reduced to single key operation.

REFERENCES

WEBSITES:

1. https://www.wikipedia.org/
2. https://www.youtube.com/
3. https://www.w3schools.com/
4. https://www.studytonight.com/c-projects/election-system-project-using-c-language
5. http://sriyncollege.org/wp-content/uploads/2021/08/ELECTRONIC-VOTING-MACHINE-DOCUMENTATION.pdf
6. https://www.javatpoint.com/c-programming-language-tutorial

BOOKS AND PAPERS

1. Balaguruswamy "Programming in C", Tata McGraw Hill 3rd Edition
2. Kanetkar, "Let us C", BPB Publications-9th edition.