Online IDE with Node JS

Submitted by

Group B

MATRIC NO	NAME
C201202	Farjana Akter Laila
C201211	Sanjida Binta Aziz
C201231	Jesmin Akter
C201234	Jakia Rahman
C201237	Tasnin Jabin

Course Code: **CSE-3528**

Course Title: **Compiler Lab**

Submitted to:

Mrs. Israt Binteh Habib

Lecturer

Dept. of CSE, IIUC

Department of Computer Science and Engineering

International Islamic University Chittagong, Bangladesh

INTRODUCTION

The Integrate Development Environment (IDE) provides an environment in which to execute specific programs written in any language. An online or web-based IDE is a compiler that allows us to compile source code and execute it online in several languages. We can store our code online in a database and access it over an internet connection by using a web-based IDE. It has always been challenging for programmers to arrange work on projects across multiple locations, especially when working in a team. Because each member of the team is required to transmit their code to each other member. The package includes a text editor and terminal system. The user would be offered the option of compiling the program in the language of their choice. The program will be compiled by the software, and the output will be returned to the user.

PROBLEM STATEMENT

It is common for the Program Developer, who produces the code for a specific project, to be responsible for the entire environment to present a specific project in front of anyone. This environment contains the software on which the code runs as well as the databases. So, even if the developer wants to test his code alone, he must have all of the necessary resources. To his rescue, we propose a Web Based IDE that solves all of his concerns. To view, alter, and test his code, the Developer only requires a web browser and an internet connection. As a result, he saves time and reduces the cost of creating an environment wherever he travels.

SOFTWARE REQUIRMENT

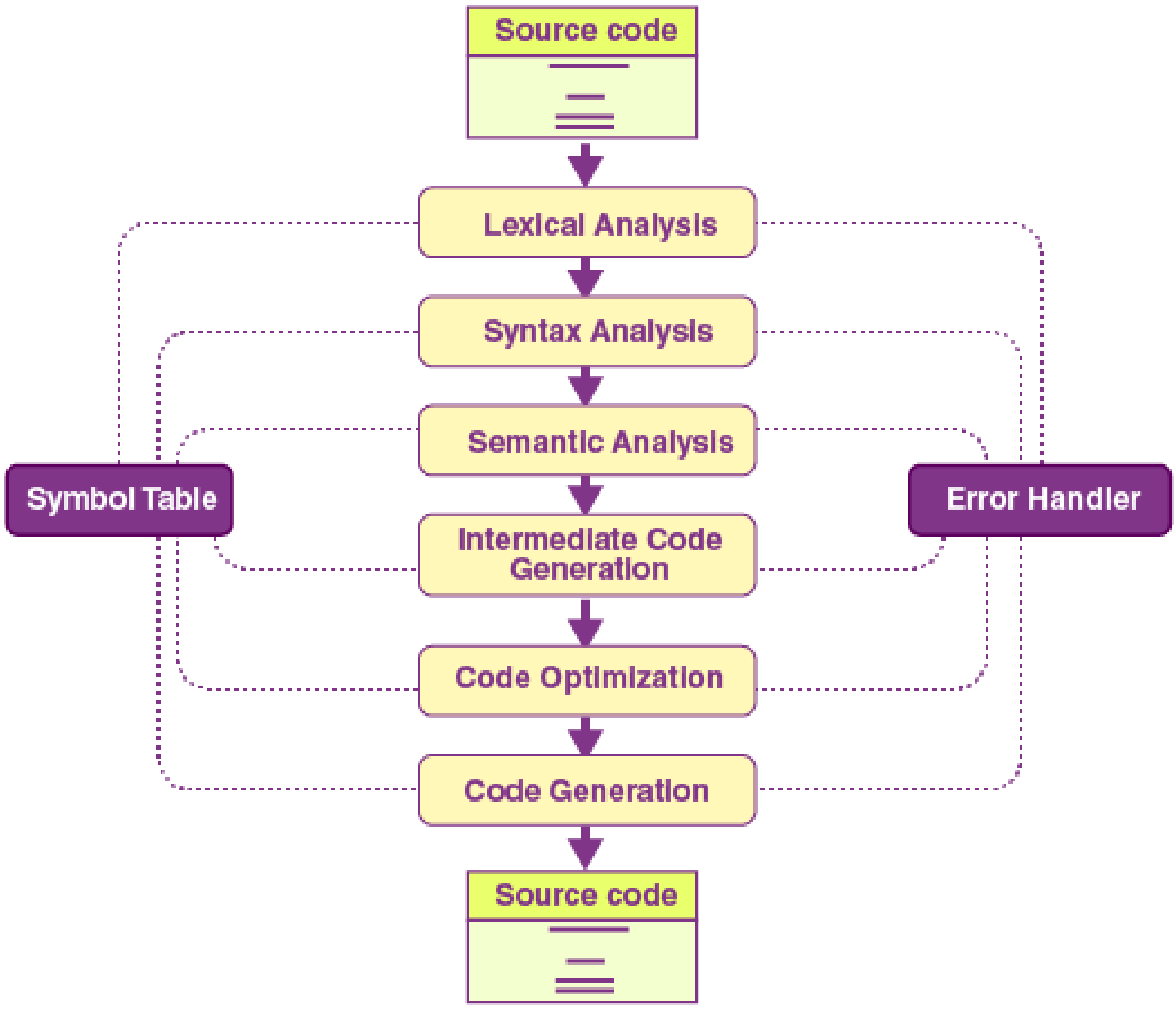
- **Node JS**

  - It allows us to use JavaScript as a backend language
  - It’s cross-platform
  - It’s lightweight
  - It simplifies developers’ work
  - It’s designed with scalability in mind
  - It offers stability and long-term support

- **VS Code:**

  - Visual Studio Code is a code editor redefined and optimized for building and debugging modern web and cloud applications.

METHODOLOGY



SYSTEM DESIGN

For our project implementation we have used waterfall methodology, one of the oldest surviving SDLC methodologies. It follows a straightforward approach, where we can completes one phase at a time, and each phase uses information from the last one to move forward.

While this methodology does make the needs and outcomes clear, and gives each stage of the model a well-defined starting and ending point, there are downsides in Waterfall's rigidity. In fact, some experts believe the Waterfall model was never meant to be a working SDLC methodology for developing software because of how fixed it is in nature. Because of this, we have applied SDLC Waterfall methods for the best implementation of our project.

SYSTEM IMPLEMENTATION

In this project development, the three important aspects need to be emphasized are JavaScript, Node JS and HTML. The software that has been used in this project is JavaScript, Node JS, HTML is used to Compile and execute the client program. It provide the IDE for client to compile their program.

- **Set up the environment for the Application:**

  First make a directory where all our files and folders will reside. Then going to terminal or cmd we will write the following commands

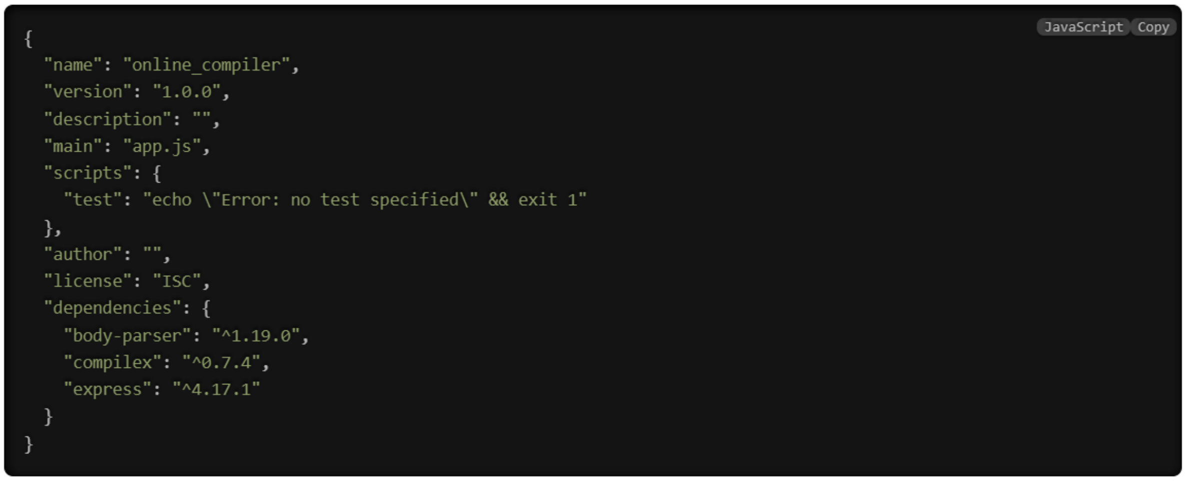
```
mkdir online_compiler
cd online_compiler
npm init //choose defaults
code .   // open visual studio code
```

- **Install the required packages:**
  Now we will install the following packages/modules that will be required for the web-application

```
npm install express --save
npm install body-parser --save
npm install compilex --save
```

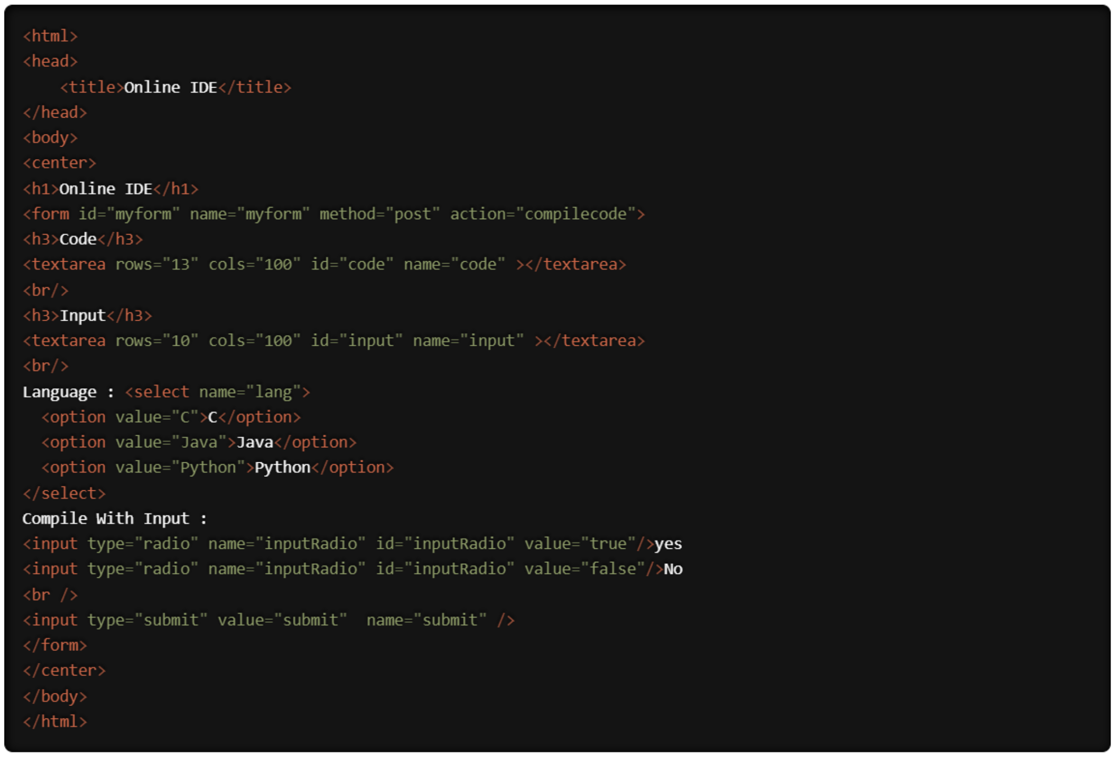
- **Let's us try to understand what these packages will be used for:**
  1-> Express will be used for routing pages and creating the server where the web application will run.
  2-> Body-parser To handle HTTP POST request in Express.js version 4 and above, we need to install middleware module called body-parser.Body-parser extract the entire body portion of an incoming request stream and exposes it on req.body.The middleware was a part of Express.js earlier but now we have to install it separately.This body-parser module parses the JSON, buffer, string and URL encoded data submitted using HTTP POST request.

3-> Compilex is a npm package which contains all the predefined functionality to create a compiler for several langauage which can be accessed using express GET and POST functions. After installation of these packages/modules the package.json should look like this:



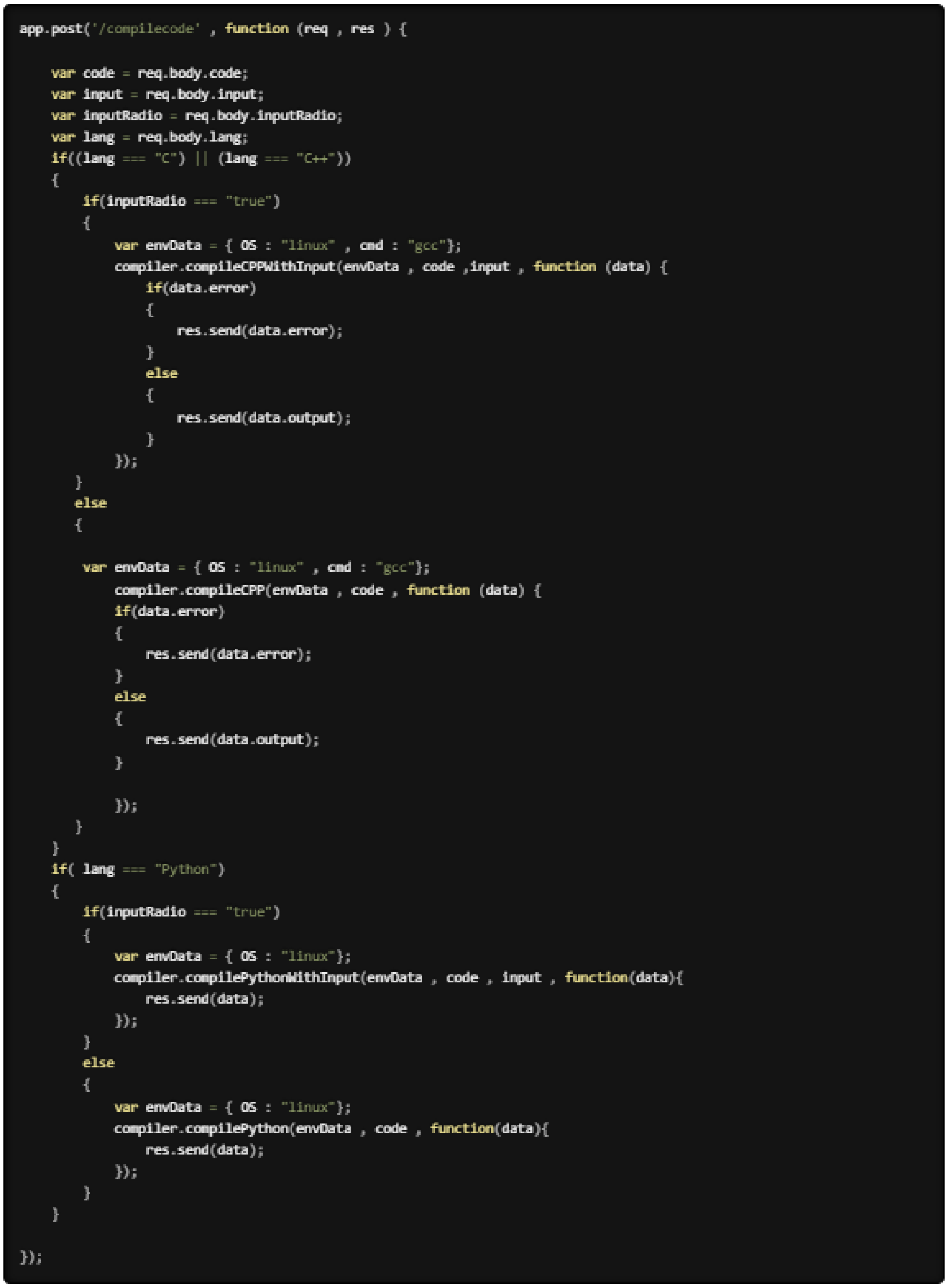
```
{
  "name": "online_compiler",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.19.0",
    "compilex": "^0.7.4",
    "express": "^4.17.1"
  }
}
```

- **Create the root page:**
  Now, we have to make a html file which will serve as our root page named index.html or but have to make the required changes in the app.js file.
  What we will do in index.html is that we will define a form which will contain textarea for writing code and input,options menu ,radio buttons for input or not respectively and submit button for result.
  index.html should look like this:



```
<html>
<head>
    <title>Online IDE</title>
</head>
<body>
<center>
<h1>Online IDE</h1>
<form id="myform" name="myform" method="post" action="compilecode">
<h3>Code</h3>
<textarea rows="13" cols="100" id="code" name="code" ></textarea>
<br/>
<h3>Input</h3>
<textarea rows="10" cols="100" id="input" name="input" ></textarea>
<br/>
Language : <select name="lang">
  <option value="C">C</option>
  <option value="Java">Java</option>
  <option value="Python">Python</option>
</select>
Compile With Input :
<input type="radio" name="inputRadio" id="inputRadio" value="true"/>yes
<input type="radio" name="inputRadio" id="inputRadio" value="false"/>No
<br />
<input type="submit" value="submit"  name="submit" />
</form>
</center>
</body>
</html>
```

- **Create index.js file:**
  We create a index.js file which will be the main page but make sure that it should be of the same name as that of the "main" in the package.json file which we should have defined during npm init process.



```
app.post('/compilecode' , function (req , res ) {

    var code = req.body.code;
    var input = req.body.input;
    var inputRadio = req.body.inputRadio;
    var lang = req.body.lang;
    if((lang === "C") || (lang === "C++"))
    {
        if(inputRadio === "true")
        {
            var envData = { OS : "linux" , cmd : "gcc"};
            compiler.compileCPPWithInput(envData , code ,input , function (data) {
                if(data.error)
                {
                    res.send(data.error);
                }
                else
                {
                    res.send(data.output);
                }
            });
       }
       else
       {

        var envData = { OS : "linux" , cmd : "gcc"};
            compiler.compileCPP(envData , code , function (data) {
            if(data.error)
            {
                res.send(data.error);
            }
            else
            {
                res.send(data.output);
            }

            });
       }
    }
    if( lang === "Python")
    {
        if(inputRadio === "true")
        {
            var envData = { OS : "linux"};
            compiler.compilePythonWithInput(envData , code , input , function(data){
                res.send(data);
            });
        }
        else
        {
            var envData = { OS : "linux"};
            compiler.compilePython(envData , code , function(data){
                res.send(data);
            });
        }
    }

});
```

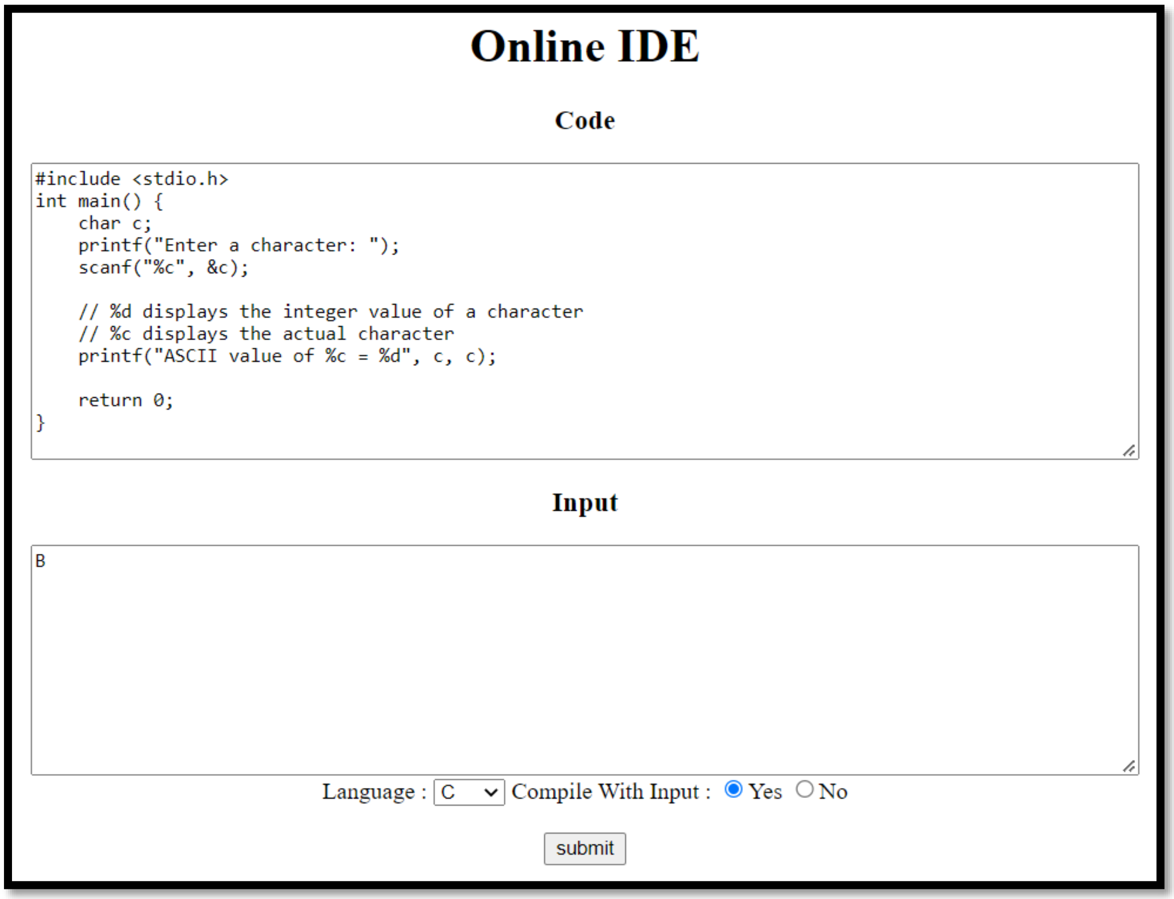
- **Summary:**
  Now what's happening here is that when we write our code in the code section and select the language of our code and also set input button if there is any and submit our code and get the output.
  That's it now run our program.

```
node app.js
```

  This will run our application.

SYSTEM RESULT

- **Interface:**



Online IDE

Code

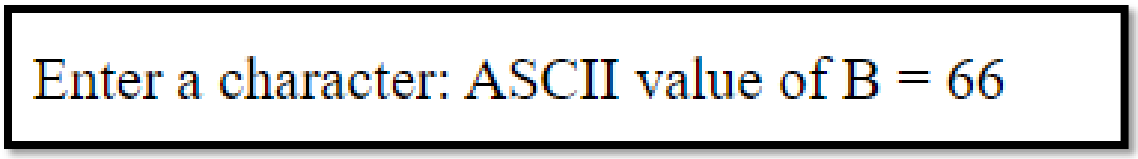
```
#include <stdio.h>
int main() {
    char c;
    printf("Enter a character: ");
    scanf("%c", &c);

    // %d displays the integer value of a character
    // %c displays the actual character
    printf("ASCII value of %c = %d", c, c);

    return 0;
}
```

Input

```
B
```

Language : C Compile With Input : [✓ Yes] No

- **Final Output:**



```
Enter a character: ASCII value of B = 66
```

FUTURE SCOPE

- Limitation of programming language.
- More functions need to be added for better interface design.
- Loading time of compiler should be faster.

CONCLUSION

The project's main goal is to provide a centralized compilation system. It will also serve as a centralized repository for all written code. The another significant advantage that this system will have over others is that it will make the user's system lightweight, eliminating the need to maintain separate compilers on the client side. Additionally, the process of maintaining and distributing dynamic usernames and passwords will be substantially streamlined. Authentication and tailored task distribution will also be available.

APPENDIX

- https://github.com/sanjidaaaziz/Online-IDE-with-Node-JS.git