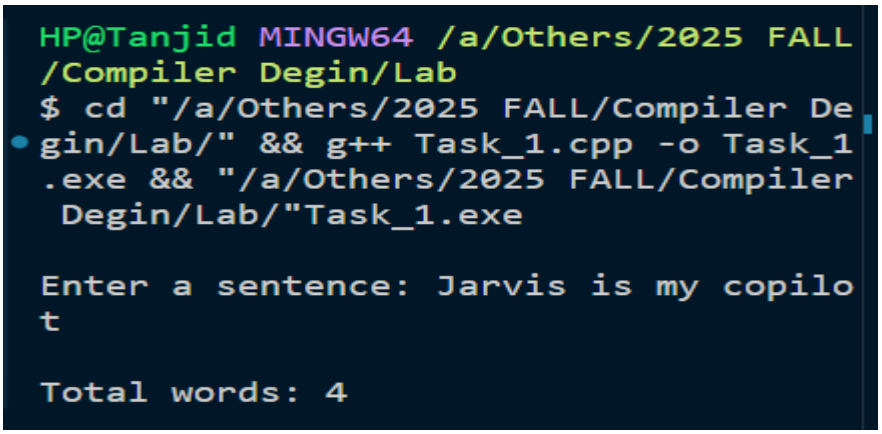
1. Write a LEX/C++ program to count word from the sentence.

Code:

```
#include <bits/stdc++.h>

using namespace std;

int main() {

string str;

cout << "Enter a sentence: ";

getline(cin, str);

stringstream ss(str);

string word;

int count = 0;

while (ss >> word) {

count++;

}

cout << "\nTotal words: " << count << endl;

return 0;

}
```

Output:



2. Write a LEX/C++ program to identify token.

Code:

```
#include <bits/stdc++.h>
using namespace std;
bool isKeyword(string s)
{
string keywords[] = {"int", "float", "if", "else", "while", "for", "return"};
int n = 7;
for (int i = 0; i < n; i++)
{
if (s == keywords[i])
return true;
}
return false;
}
bool isNumber(string s)
{
bool hasDecimal = false;
for (char c : s)
{
if (c == '.')
{
if (hasDecimal)
return false;
```

```
hasDecimal = true;
}
else if (!isdigit(c))
return false;
}
return !s.empty();
}
bool isIdentifier(string s)
{
if (s.empty())
return false;
if (!isalpha(s[0]) && s[0] != '_')
return false;
for (char c : s)
{
if (!isalnum(c) && c != '_')
return false;
}
return true;
}
int main()
{
string str = "int main() { int a = 10; float b = 3.14; if (a > b) { return 0; } }";
string token = "";
for (int i = 0; i < str.size(); i++)
```

```
{
char c = str[i];
if (isspace(c) || ispunct(c))
{
if (!token.empty())
{
if (isKeyword(token))
cout << token << " -> Keyword\n";
else if (isNumber(token))
cout << token << " -> Number\n";
else if (isIdentifier(token))
cout << token << " -> Identifier\n";
else
cout << token << " -> Unknown\n";
token.clear();
}
if (ispunct(c) && c != ' ')
{
cout << c << " -> Operator/Symbol\n";
}
}
else
{
token += c;
}
```

```
}
if (!token.empty())
{
if (isKeyword(token))
cout << token << " -> Keyword\n";
else if (isNumber(token))
cout << token << " -> Number\n";
else if (isIdentifier(token))
cout << token << " -> Identifier\n";
else
cout << token << " -> Unknown\n";
}
return 0;
}
```

Output:

```
HP@Tanjid MINGW64 /a/Others/2025 FALL
/Compiler Degin/Lab
$ cd "/a/Others/2025 FALL/Compiler De
gin/Lab/" && g++ Task_2.cpp -o Task_2
.exe && "/a/Others/2025 FALL/Compiler
 Degin/Lab/"Task_2.exe
int -> Keyword
main -> Identifier
( -> Operator/Symbol
) -> Operator/Symbol
{ -> Operator/Symbol
int -> Keyword
a -> Identifier
= -> Operator/Symbol
10 -> Number
; -> Operator/Symbol
float -> Keyword
b -> Identifier
= -> Operator/Symbol
3 -> Number
. -> Operator/Symbol
14 -> Number
; -> Operator/Symbol
if -> Keyword
( -> Operator/Symbol
a -> Identifier
> -> Operator/Symbol
b -> Identifier
) -> Operator/Symbol
{ -> Operator/Symbol
return -> Keyword
0 -> Number
; -> Operator/Symbol
} -> Operator/Symbol
} -> Operator/Symbol
```

3. Write a C++ program to convert an Infix to Postfix/Prefix Expression using stack.

Code:

```
#include <bits/stdc++.h>

using namespace std;
```

```
int precedence(char op)
{
if (op == '+' || op == '-')
return 1;
if (op == '*' || op == '/')
return 2;
return 0;
}
string infixToPostfix(string infix)
{
stack<char> st;
string postfix = "";
for (char ch : infix)
{
if (isalnum(ch))
{
postfix += ch;
}
else if (ch == '(')
{
st.push(ch);
}
else if (ch == ')')
{
while (!st.empty() && st.top() != '('
```

```
{
postfix += st.top();
st.pop();
}
st.pop();
}
else
{
while (!st.empty() && precedence(st.top()) >= precedence(ch))
{
postfix += st.top();
st.pop();
}
st.push(ch);
}
}
while (!st.empty())
{
postfix += st.top();
st.pop();
}
return postfix;
}
int main()
{
```

```
string infix;
infix = "((A+B)*C)";
string postfix = infixToPostfix(infix);
cout << "Postfix Expression:" << "\n"
<< postfix << endl;
return 0;
}
```

Output:

```
HP@Tanjid MINGW64 /a/Others/2025 FALL/
Compiler Degin/Lab
$ cd "/a/Others/2025 FALL/Compiler Deg
in/Lab/" && g++ Task_3.cpp -o Task_3.e
xe && "/a/Others/2025 FALL/Compiler De
gin/Lab/"Task_3.exe

Postfix Expression: AB+C*
```

4. Write a C/C++ program to identify whether a given line is a comment or not.

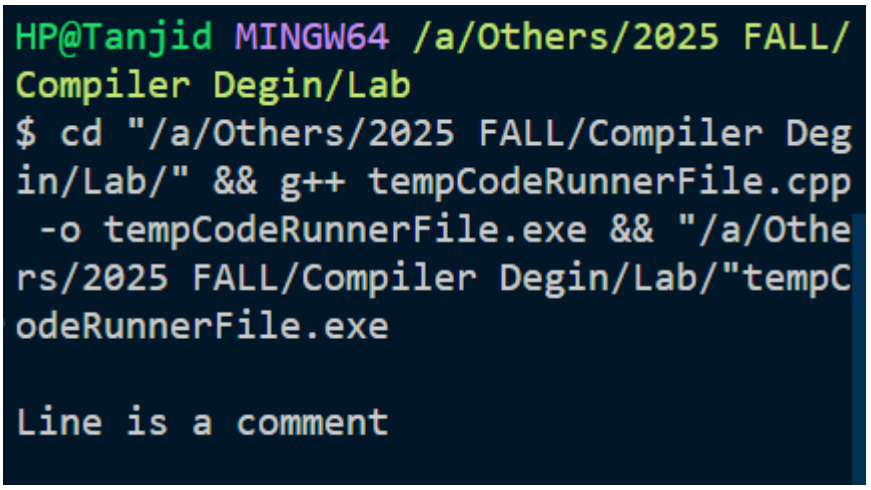
Code:

```
#include <bits/stdc++.h>
using namespace std;
bool isComment(string str)
{
int length = str.length();
char firstChar = str[0];
```

```
char secondChar = str[1];

char lastChar = str[str.length() - 1];

char secondLastChar = str[str.length() - 2];

if (length >= 2 && firstChar == '/' && secondChar == '/')

{

return true;

}

if (length >= 4 && firstChar == '/' && secondChar == '*' &&
secondLastChar == '*' &&

lastChar == '/')

{

return true;

}

return false;

}

int main()

{

string s;

// cout << "Enter a string: ";

// getline(cin, s);

s = "/*GeeksForGeeks GeeksForGeeks*/";

if (isComment(s))

{

cout << "\nLine is a comment" << endl;

}
```

```
else
{
cout << "\nLine is not a comment" << endl;
}
return 0;
}
```

Output:



5. Write a C/C++ program to identify valid keywords.

Code:

```
#include <bits/stdc++.h>
using namespace std;
string keywords[] = {
"alignas", "alignof", "and", "and_eq", "asm", "atomic_cancel",
"atomic_commit", "atomic_noexcept",
```

"auto", "bitand", "bitor", "bool", "break", "case", "catch", "char", "char8_t", "char16_t",

"char32_t", "class", "compl", "concept", "const", "consteval", "constexpr", "constinit",

"const_cast", "continue", "co_await", "co_return", "co_yield", "decltype", "default",

"delete", "do", "double", "dynamic_cast", "else", "enum", "explicit", "export", "extern",

"false", "float", "for", "friend", "goto", "if", "inline", "int", "long", "mutable",

"namespace", "new", "noexcept", "not", "not_eq", "nullptr", "operator",

"or", "or_eq", "private",

"protected", "public",

"register",

"reinterpret_cast",

"requires",

"return",

"short",

"signed",

"sizeof",

"static",

"static_assert",

"static_cast",

"struct",

"switch",

"synchronized",

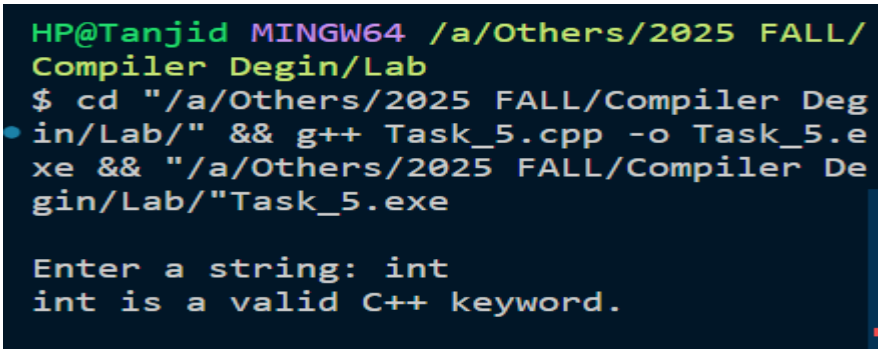
"template",

"this",

```
"thread_local",
"throw",
"true",
"try",
"typedef",
"typeid",
"typename",
"union",
"unsigned",
"using",
"virtual",
"void",
"volatile",
"wchar_t",
"while",
"xor",
"xor_eq"};
bool isKeyword(const string &word)
{
for (const string &keyword : keywords)
{
if (word == keyword)
{
return true;
}
```

```
}
return false;
}
int main()
{
cout << "Enter a string: ";
string s;
cin >> s;
if (isKeyword(s))
{
cout << s << " is a valid C++ keyword." << endl;
}
else
{
cout << s << " is not a valid C++ keyword." << endl;
}
return 0;
}
```

Output:



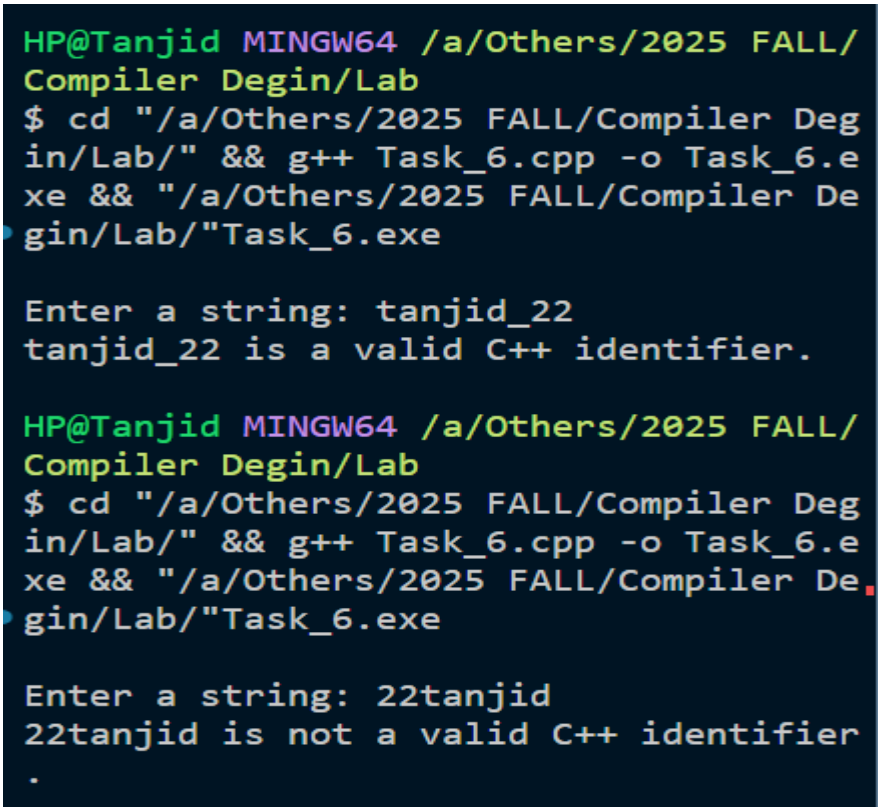
6. Write a C/C++ program to recognize valid and invalid identifier.

Code:

```
#include <bits/stdc++.h>
using namespace std;
bool isValidIdentifier(string str)
{
if (str.empty())
return false;
if (!((str[0] >= 'a' && str[0] <= 'z') ||
(str[0] >= 'A' && str[0] <= 'Z') ||
str[0] == '_'))
{
return false;
}
for (int i = 1; i < str.length(); i++)
{
if (isspace(str[i]))
return false;
if (!((str[i] >= 'a' && str[i] <= 'z') ||
(str[i] >= 'A' && str[i] <= 'Z') ||
(str[i] >= '0' && str[i] <= '9') ||
str[i] == '_'))
{
return false;
```

```
}
}
return true;
}
int main()
{
cout << "\nEnter a string: ";
string s;
getline(cin, s);
if (isValidIdentifier(s))
cout << s << " is a valid C++ identifier." << endl;
else
cout << s << " is not a valid C++ identifier." << endl;
return 0;
}
```

Output:


HP@Tanjid MINGW64 /a/Others/2025 FALL/Compiler Degin/Lab
$ cd "/a/Others/2025 FALL/Compiler Degin/Lab/" && g++ Task_6.cpp -o Task_6.exe && "/a/Others/2025 FALL/Compiler Degin/Lab/"Task_6.exe
Enter a string: tanjid_22
tanjid_22 is a valid C++ identifier.
HP@Tanjid MINGW64 /a/Others/2025 FALL/Compiler Degin/Lab
$ cd "/a/Others/2025 FALL/Compiler Degin/Lab/" && g++ Task_6.cpp -o Task_6.exe && "/a/Others/2025 FALL/Compiler Degin/Lab/"Task_6.exe
Enter a string: 22tanjid
22tanjid is not a valid C++ identifier.