Name- **LAB-ASSIGNMENT** **16JE00 …..**

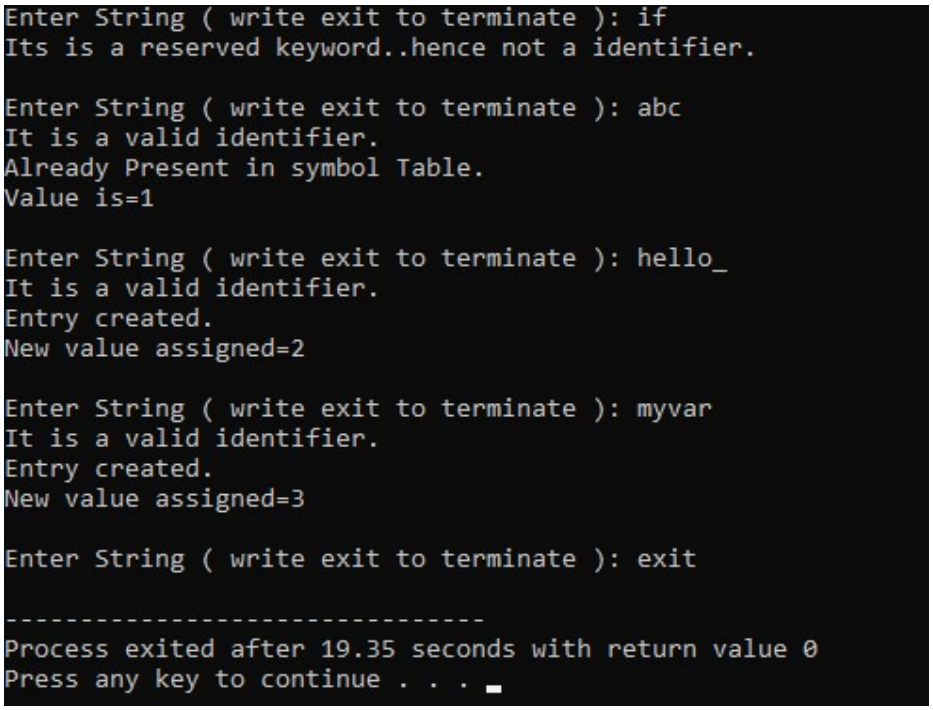
Q1. WAP to check whether a given identifier is valid or not . Create a symbol table for the same.

//Code

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;

ll x=1;

bool check(string s)
{
   if( !( s[0]=='_' || (s[0]>=65 && s[0]<=90) || (s[0]>=97 && s[0]<=122)) )
      return 0;

   for(int i=0;i<s.length();i++)
      if(!((s[i]>=65 && s[i]<=90) || (s[i]>=97 && s[i]<=122) || s[i]=='_' || (s[i]>=48 && s[i]<=57)))
         return 0;
   return 1;
}

map<string,ll> loadMap()
{
   map<string,ll> m;
   ifstream fin;
   fin.open("table_of_identifier.txt");
   ll ind;
   string s;
   while((fin>>s)&&(fin>>ind))
   {
         m[s]=ind;
         x=ind+1;
         }
   fin.close();
   return m;
}

int main()
{
   string strings[] =
{"auto","break","case","char","const","continue","default","do","double","else","enum","extern","float","for",

"goto","if","int","long","register","return","short","signed","sizeof","sattic","struct","switch","typedef","union",
                  "unsigned","void","volatile","while"};
   set<string> keywords;
   for(int i=0;i<32;i++)
      keywords.insert(strings[i]);
   map<string,ll> symbolTable=loadMap();
   string s;
   ofstream fout;
   fout.open("table_of_identifier.txt",std::ios_base::app);
   while(1)
   {
      cout<<"Enter String ( write exit to terminate ): ";
      cin>>s;
      if(s=="exit")
          break;
      if(keywords.find(s)!=keywords.end())
```

```
        {
            cout<<"Its is a reserved keyword..hence not a identifier.\n\n";
            continue;
        }
        if(check(s))
        {
            cout<<"It is a valid identifier.\n";
            if(symbolTable[s]==0)
            {
                symbolTable[s]=x++;
                fout<<s<<" "<<x-1<<endl;
                cout<<"Entry created.\nNew value assigned="<<symbolTable[s]<<endl;
            }
            else
                cout<<"Already Present in symbol Table.\nValue is="<<symbolTable[s]<<endl;
        }
        else
            cout<<"Invalid Identifier.\n";
        cout<<endl;
    }
    fout.close();
}
```

OUTPUT:



Q2. WAP to check whether a given keyword is valid or not . Create a symbol table for the same.

//Code

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;

bool check(string s)
{
   if( !( s[0]=='_' || (s[0]>=65 && s[0]<=90) || (s[0]>=97 && s[0]<=122)) )
      return 0;
   for(int i=0;i<s.length();i++)
      if(!((s[i]>=65 && s[i]<=90) || (s[i]>=97 && s[i]<=122) || s[i]=='_' || (s[i]>=48 && s[i]<=57)))
         return 0;
   return 1;
}

map<string,ll> loadMap()
{
   map<string,ll> m;
   ifstream fin;
   fin.open("list_of_keyword.txt");
   ll ind;
   string s;
   while((fin>>s)&&(fin>>ind))
      m[s]=ind;
   fin.close();
   return m;
}

int main()
{
         string strings[] = {"auto","break","case","char"," const","continue","default",
"else","enum","extern","float","for","goto","if","int","long","register","return","short","signed",
"sizeof","sattic","struct","switch","typedef","union", "unsigned","void","volatile","while"};

   set<string> keywords;
   for(int i=0;i<32;i++)
      keywords.insert(strings[i]);

   ifstream fin;
   fin.open("list_of_keywords.txt");
   map<string,ll> symbolTable;
   string ch;
   ofstream fout;
   fout.open("list_of_keyword.txt",std::ios_base::app);
   if(!(fin>>ch))
      for(int i=0;i<32;i++)
         fout<<strings[i]<<" "<<i+1<<endl;

   fin.close();
   symbolTable=loadMap();
   string s;
```

```
int x=33;
while(1)
{
   cout<<"Enter String ( write exit to terminate ): ";
   cin>>s;
   if(s=="exit")
        break;
   if(keywords.find(s)!=keywords.end())
   {
      cout<<"Its is a reserved keyword.\nValue is = "<<symbolTable[s]<<endl<<endl;
      continue;
   }
   if(check(s))
   {
      cout<<"It is a valid identifier..not a keyword.\n";
      if(symbolTable[s]==0)
      {
         symbolTable[s]=x++;
         fout<<s<<" "<<x-1<<endl;
         cout<<"Entry created.\nNew value assigned="<<symbolTable[s]<<endl;
      }
      else
         cout<<"Already Present in symbol Table.\nValue is="<<symbolTable[s]<<endl;
   }
   else
      cout<<"Neither Keyword nor Identifier.\n";
   cout<<endl;
}
fout.close();}
```

OUTPUT:

```
Enter String ( write exit to terminate ): if
Its is a reserved keyword.
Value is = 16

Enter String ( write exit to terminate ): auto
Its is a reserved keyword.
Value is = 1

Enter String ( write exit to terminate ): abc
It is a valid identifier..not a keyword.
Entry created.
New value assigned=33

Enter String ( write exit to terminate ): exit

--------------------------------
Process exited after 24.64 seconds with return value 0
Press any key to continue . . .
```

Q3. WAP to check whether a Given Operator is valid or not . Create a symbol table for the same.

//Code

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
ll x=1;
map<string,pair<ll,string> > loadMap()
{
   map<string,pair<ll,string> > m;
   ifstream fin;
   fin.open("operators_table.txt");
   ll ind;
   string op,name;
   while((fin>>op)&&(fin>>ind)&&(fin>>name))
   {
         m[op].first=ind;
         x=ind+1;
         m[op].second=name;
         }
   fin.close();
   return m;
}

int main()
{
   map<string,pair<ll,string> > symbolTable=loadMap();
   string s;
   ofstream fout;
   fout.open("operators_table.txt",std::ios_base::app);
   while(1)
   {
      cout<<"Enter Operator (write exit to terminate): ";
      cin>>s;
      if(s=="exit")
          break;
      if(symbolTable[s].first>0)
      {
          cout<<symbolTable[s].second<<endl;
          cout<<"Already Present in symbol Table.\nIndex Value is="<<symbolTable[s].first<<endl<<endl;
          continue;
                  }
      if(s.length()>=3)
      {
          cout<<"Invalid Operator.\n\n";
          continue;
                  }
                  bool valid=0;
                  string res="";
      switch(s[0])
      {
          case '+':
                  if(s.length()==1)
```

```
		{
			valid=1;
			res="Adition_";
				}
		else if(s[1]=='+')
		{
			valid=1;
			res="Increment_";
				}
				else if(s[1]=='=')
				{
					valid=1;
					res="Addition_Assignment_";
				}
		break;
case '-':
		if(s.length()==1)
		{
			valid=1;
			res="Subtraction_";
				}
		else if(s[1]=='-')
		{
			valid=1;
			res="Decrement_";
				}
				else if(s[1]=='=')
				{
					valid=1;
					res="Subtraction_Assignment_";
				}
		break;
case '*':
		if(s.length()==1)
		{
			valid=1;
			res="Multiplication_";
				}
		else if(s[1]=='=')
		{
			valid=1;
			res="Multiplication_Assignment_";
				}
		break;
case '/':
		if(s.length()==1)
		{
			valid=1;
			res="Division_";
				}
		else if(s[1]=='=')
		{
			valid=1;
			res="Division_Assignment_";
				}
```

```
		break;
	case '%':
		if(s.length()==1)
		{
			valid=1;
			res="Modulus_";
				}
		else if(s[1]=='=')
		{
			valid=1;
			res="Modulus_Assignment_";
				}
		break;
	case '=':
		if(s.length()==1)
		{
			valid=1;
			res="Assignment_";
				}
		else if(s[1]=='=')
		{
			valid=1;
			res="Comparision_";
				}
		break;
	case '>':
		if(s.length()==1)
		{
			valid=1;
			res="Greater_than_";
				}
		else if(s[1]=='=')
		{
			valid=1;
			res="Greater_than_or_equal_to_";
				}
		break;
	case '<':
		if(s.length()==1)
		{
			valid=1;
			res="Less_than_";
				}
		else if(s[1]=='=')
		{
			valid=1;
			res="Less_than_or_equal_to_";
				}
		break;
	case '!':
		if(s.length()==1)
		{
			valid=1;
			res="Negation_";
				}
```

```
                else if(s[1]=='=')
                {
                        valid=1;
                        res="Not_equal_to_";
                                }
                break;
         default:
                cout<<"Invalid Operator.\n";
                }
                if(!valid)
                        cout<<"Invalid Operator.\n";
      else
      {
          res=res+"Operator.";
        cout<<res<<endl;
        if(symbolTable[s].first==0)
        {
           symbolTable[s].first=x++;
           symbolTable[s].second=res;
           fout<<s<<"\t"<<x-1<<"\t"<<res<<endl;
           cout<<"Entry created.\nNew Index value assigned="<<symbolTable[s].first<<endl;
        }
        else
           cout<<"Already Present in symbol Table.\nIndex Value is="<<symbolTable[s].first<<endl;
      }
      cout<<endl;
   }
   fout.close();
}
```

OUTPUT:

```
Adition_Operator.
Entry created.
New Index value assigned=2

Enter Operator (write exit to terminate): -=
Subtraction_Assignment_Operator.
Entry created.
New Index value assigned=3

Enter Operator (write exit to terminate): %
Modulus_Operator.
Entry created.
New Index value assigned=4

Enter Operator (write exit to terminate): !=
Not_equal_to_Operator.
Entry created.
New Index value assigned=5

Enter Operator (write exit to terminate): >=
Greater_than_or_equal_to_Operator.
Entry created.
New Index value assigned=6

Enter Operator (write exit to terminate): =
Assignment_Operator.
Entry created.
New Index value assigned=7

Enter Operator (write exit to terminate): exit

--------------------------------
Process exited after 15.94 seconds with return value 0
Press any key to continue . . .
```