……1.h……..

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

// E -> iF
// F -> +F / t

void match(string input, int &index, char ch);
void E(string input, int &index, bool &invalid);
void F(string input, int &index, bool &invalid);
```

……..1.cpp……..

```
#include "1.h"

void E(string input, int& index, bool &invalid)
{
   if (input[index] == 'i')
   {
      index++;
      F(input, index, invalid);
   }
}

void F(string input, int& index, bool &invalid)
{
   char element = input[index];

   if (element == '+')
   {
      index++;

      if (input[index] == 'i')
      {
         index++;
         F(input, index, invalid);
      }
      else
      {
         cout << "Invalid Pattern" << endl;
         invalid = true;
         return;
      }
   }
   else
      return;
}
```

……1_m.cpp……..

```
#include "1.h"

int main()
```

```
{
    string input;// = "i+i$";
    int index = 0;
    bool invalid = false;

    cout << "Enter the input string: ";
    getline(cin, input);

    input += '$';

    E(input, index, invalid);

    if (!invalid && input[index] == '$')
        cout << "Parsing Successful" << endl;
    else
        cout << "Unsuccessful" << endl;

    return 0;
}
```

```
pawan@pawan:~/Desktop/Code/CD/Recursive_Descent_Parser/1$ ./m.out
Enter the input string: ^C
pawan@pawan:~/Desktop/Code/CD/Recursive_Descent_Parser/1$ g++ 1.cpp 1_m.cpp -o m.out
pawan@pawan:~/Desktop/Code/CD/Recursive_Descent_Parser/1$ ./m.out
Enter the input string: i+
Invalid Pattern
Unsuccessful
pawan@pawan:~/Desktop/Code/CD/Recursive_Descent_Parser/1$ ./m.out
Enter the input string: i+
Invalid Pattern
Unsuccessful
pawan@pawan:~/Desktop/Code/CD/Recursive_Descent_Parser/1$ ./m.out
Enter the input string: i+i+
Invalid Pattern
Unsuccessful
pawan@pawan:~/Desktop/Code/CD/Recursive_Descent_Parser/1$ ./m.out
Enter the input string: i
Parsing Successful
pawan@pawan:~/Desktop/Code/CD/Recursive_Descent_Parser/1$
```

……2.h……

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

        // E -> TEdash
        // Edash -> +TEdash/#
        // T -> FTdash
        // Tdash -> *FTdash/#
        // F -> (E)/id
```

```
void E(string input, int &index, bool &valid);
void Edash(string input, int &index, bool &valid);
void T(string input, int &index, bool &valid);
void Tdash(string input, int &index, bool &valid);
void F(string input, int &index, bool &valid);
```

……..2.cpp……

```
#include "2.h"

void E(string input, int &index, bool &valid)
{
   T(input, index, valid);
   Edash(input, index, valid);
}


void Edash(string input, int &index, bool &valid)
{
   if (input[index] == '+')
   {
      index++;
      T(input, index, valid);
      Edash(input, index, valid);
   }
   else
      return;
}


void T(string input, int &index, bool &valid)
{
   F(input, index, valid);
   Tdash(input, index, valid);
}


void Tdash(string input, int &index, bool &valid)
{
   if (input[index] == '*')
   {
      index++;
      F(input, index, valid);
      Tdash(input, index, valid);
   }
   else
      return;
}


void F(string input, int &index, bool &valid)
{
```

```
    if (input[index] == '(')
    {
        index++;
        E(input, index, valid);

        if (input[index] == ')')
        {
            index++;
            return;
        }
    }
    else if (input[index] == 't')
    {
        index++;
        return;
    }
    else
    {
        cout << "Invalid Input String." << endl;
        valid = false;
        return;
    }


}
```

……..2_m.cpp……..

```
#include "2.h"

int main()
{
    string input;
    int index = 0;
    bool valid = true;

    cout << "Enter the input string: ";
    getline(cin, input);

    input += "$";

    cout << endl << input << endl;

    E(input, index, valid);

    if (valid && input[index] == '$')
        cout << "Parsing Successful" << endl;
    else
        cout << "Unsuccessful" << endl;
    return 0;
}
```

```
pawan@pawan:~/Desktop/Code/CD/Recursive_Descent_Parser/2$ g++ 2.cpp 2_m.cpp -o t.out
pawan@pawan:~/Desktop/Code/CD/Recursive_Descent_Parser/2$ ./t.out
Enter the input string: (t)

(t)$
Parsing Successful
pawan@pawan:~/Desktop/Code/CD/Recursive_Descent_Parser/2$ ./t.out
Enter the input string: (+)

(+)$
Invalid Input String.
 Unsuccessful
```

……SHUBHAM……

```
Enter the Variable to find First and Follow Set: S

The given Grammar is:
A -> f
S -> ASe / tSd


The First Set of S is: t f

The Follow Set of S is: d e $
```

……..……1……..

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

void E(string   str, int &i );
void F(string   str, int &i );

void E(string str, int& i)
{
   if (str[i] == 'i')
   {
      i++;
      F(str, i);
   }
}

void F(string str, int& i)
{
   char element = str[i];

   if (element == '+')
```

```
    {
        i++;

        if (str[i] == 'i')
        {
            i++;
            F(str, i  );
        }
    }
    else
        return;
}


int main()
{
    string str;
    int i = 0;

    cout << "Enter string";
    getline(cin, str);

    str += '$';

    E(str, i);

    if (str[i] == '$')
        cout << "Parsing Successful" << endl;
    else
        cout << "Unsuccessful" << endl;

    return 0;
}
```

```
Enter string i+i
successful parsing
```

```
Enter string i++
unsuccessful parsing
```

……..2……..

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

void E(string input, int &index );
void Edash(string input, int &index );
void T(string input, int &index );
void Tdash(string input, int &index );
void F(string input, int &index );
```

```
void E(string  str, int &i )
{
   T( str, i );
   Edash( str, i );
}


void Edash(string  str, int &i )
{
   if ( str[i] == '+')
   {
      i++;
      T( str, i );
      Edash( str, i );
   }
   else
      return;
}


void T(string  str, int &i )
{
   F( str, i );
   Tdash( str, i );
}


void Tdash(string  str, int &i )
{
   if ( str[i] == '*')
   {
      i++;
      F( str, i );
      Tdash( str, i );
   }
   else
      return;
}


void F(string  str, int &i )
{
   if ( str[i] == '(')
   {
      i++;
      E( str, i );

      if ( str[i] == ')')
      {
         i++;
```

```
            return;
        }
    }
    else if ( str[i] == 'v')
    {
        i++;
        return;
    }
}



int main()
{
    string str;
    int i = 0;
    cout << "enter string ";
    getline(cin, str);
    str += "$";
    E(str, i);
    if (str[i] == '$')
        cout << "successful parsing" << endl;
    else
        cout << "unsuccessful parsing" << endl;
    return 0;
}
```

```
enter string (v)
successful parsing
```

```
enter string ()v
unsuccessful parsing
```