ID	属性	優先度	要求事項
A 機能要求（条件処理表の作成）			
A-01	要求	高	gccで構文解析ができ，相互に依存関係がない1つ以上のC言語ソースファイルが与えられたとき，条件処理表を作成できる
	補足		相互に依存関係がある複数のC言語ソースファイルが与えられた場合は，それぞれの依存関係のないファイルとして扱う gccによる構文解析ができるか判断する上で，以下の項目は考慮しないこととする ・変数・関数宣言の有無 ・ヘッダファイルのincludeの有無 gccのバージョンは，MinGW32-gcc 5.3.0を想定する.
A-01-01	要求	中	解析対象ファイルを置くフォルダにC言語ソースファイルが存在しない場合は，条件処理表の作成を中止し，エラーダイアログを表示する
A-01-02	要求	高	解析対象ファイルを置くフォルダにヘッダファイルが存在する場合は，includeをするかユーザに問い合わせる
A-01-02-01	要求	高	ユーザがヘッダファイルのincludeを許可するならば，すべてのヘッダファイルをgccによってマクロ展開し，条件処理表を作成する
A-01-03	要求	高	C言語ソースファイルではないファイルが与えられた場合は，条件処理表の作成を中止し，エラーダイアログを表示する
A-01-04	要求	中	相互に依存関係がある複数のソースファイルが与えられた場合は，警告ダイアログを表示した上で，それぞれを依存関係のないファイルとして扱う
A-01-05	要求	中	gccで構文解析ができない場合は，条件処理表の作成を中止し，エラーダイアログを表示する
A-02	要求	高	ソースコードに存在するif構文を条件として抽出できる
	補足		if構文には，複数の条件分岐(if…else if…else)を含む 例） if(a==0) {b = 1;} else if (a==1) {b = 2;} else {b = 3;}
A-03	要求	高	ソースコードに存在するswitch構文を条件として抽出できる
	補足		switch構文には，1つ以上のcase文を含む switch構文に，break文，default文が含まれるかどうかは問わない 例） switch(a){ case 0: b = 1; case 1:b = 2; default: b = 3; } , switch(a){ case 0: b = 1; break; case 1:b = 2;} のどちらも条件として抽出できる.
A-03-01	要求	高	ソースコードにswitch構文が存在する場合，そのswitch構文と同じ制御構造を表すif構文に置き換えて条件処理表に記述する
A-04	要求	高	ソースコードに存在するif構文，switch構文の組合せを条件として抽出できる
	補足		if構文，switch構文とは，それぞれA-02,A-03に定義される構文を指す
A-04-01	要求	中	ネストの深さが高々5であるソースコードを対象に，条件処理表を作成できる
A-05	要求	高	if構文，switch構文の直前・直後にある処理を抽出できる
A-06	要求	中	条件処理表作成のための実行処理が，操作開始後60秒以内で終了しない場合は，60秒が経過した時点で処理を中止してダイアログを表示する
	補足		実行処理後，Excelシート上に表の作成が完了するまでの時間は60秒を超えてもよい.
A-07	要求	低	複雑度 最大260のソースコードを対象に，条件処理表を作成できる
	補足		複雑度260とは，あるOSSの複雑度を調査したところで，もっとも高かった値である. 本ツールが対象とするレガシーコードの中には，非常に高い複雑度のものもあると考えられるため，この値を基準とした.
A-07-01	要求	低	複雑度が260を超える場合は，警告のダイアログを表示して，出力される結果が正しいことを保証できないことを知らせる
B 機能要求（状態遷移表作成）			
B-01	要求	中	状態変数選択のフォームにおいて，ユーザが検索した状態変数が使われているソースコードの行をハイライト表示できる
B-02	要求	中	状態変数選択のフォームにおいて，ユーザはそれぞれの状態変数にコメント文を付与できる
B-02-01	要求	中	状態変数選択のフォームにおいて，状態変数名とコメント文の一覧をファイルとして外部出力できる
B-03	要求	高	条件処理表が作成でき，状態遷移設計されている関数に対して，ユーザが選択した状態変数に関する状態遷移表を作成できる
B-03	補足		「状態遷移設計されている関数」とは，状態変数が存在する関数を指す. 「状態変数」とは，以下の(a-1)(a-2)(a-3)を満たす変数を指す (a-1) ある構文の条件式で，値を比較される※単一の変数である (a-2) 関数内で取りうる状態変数の値の種類は最大20である (a-3) (a-1)の構文の内部で更新される変数である ※条件式内で比較演算子が使われず，変数のみが指定される場合も含む
B-03	補足		状態変数を含む条件式が，関数外に存在する場合は，その条件式に関する状態遷移表は作成されない.
B-03-01	要求	中	ユーザが選択した状態変数のとる有限値の種類が20を超える関数については，状態遷移表を作成しない.
B-03-02	要求	中	ユーザが選択した状態変数が含まれるif構文，switch構文のネストの深さは高々5とする
B-03-03	要求	高	条件式内で状態変数とあわせて使われる比較演算子は，"=="とする. "=="以外の比較演算子が使われる関数については，状態遷移表を作成しない.
B-03-03	補足		switch構文の場合は，変数の値とcase文で指定された定数の値を "==" で比較しているとみなす.
B-03-03	補足		今後，"=="以外の比較演算子が使われる関数も，状態遷移表作成の対象に含むことを検討している.

<table>
<tr><td>B-03-04</td><td>要求</td><td>中</td><td>B-03-01, B-03-03の要求に基づき, 状態遷移表作成の対象とならない関数がある場合, すべての関数に対して状態遷移表作成の処理が終了した後, エラーダイアログを表示する. これにより, どの関数で状態遷移表が作成されなかったかユーザに示す.</td></tr>
<tr><td>B-03-04</td><td>補足</td><td></td><td>B-03-03の場合, 比較演算子"=="のみが使われる関数は状態遷移表作成の対象となる.
"=="以外の比較演算子が使われる関数は状態遷移表作成の対象とならず, エラーダイアログを表示する.</td></tr>
<tr><td>B-03-04</td><td>補足</td><td></td><td>エラーダイアログでは, 発生したエラーの種類によって異なるエラーメッセージを表示する.</td></tr>
<tr><td>B-03-05</td><td>要求</td><td>中</td><td>想定しないエラーが発生した場合, 処理のログを取得し, ファイル出力する.
その後, エラーが発生した関数に対する状態遷移表作成の処理を中断して, 次の関数に対する処理に移行する.</td></tr>
<tr><td>B-04</td><td>要求</td><td>高</td><td>作成される状態遷移表の読み方と書式は, 要求仕様書の「2.5 状態遷移表の読み方」および「2.6 状態遷移表の書式」に準ずる</td></tr>
<tr><td>B-04-01</td><td>要求</td><td>高</td><td>状態遷移表の各列の幅は20ポイントとする. セル内の文字列の長さが列の幅を超える場合, 文字の折り返しはされずに表記される.</td></tr>
<tr><td>B-04-01-01</td><td>要求</td><td>高</td><td>if構文のブロックが表記されているセルに限り, 文字の折り返しをする.</td></tr>
<tr><td>B-05</td><td>要求</td><td>高</td><td>ユーザが選択した状態変数が条件式に存在するif構文,switch構文（以下, 制御構文Aとする）があり, かつ, 制御構文Aを含むネストにif構文, switch構文（以下, 制御構文Bとする）がある場合, 状態遷移表において制御構文Aの処理が出力されるセルと同じ行に, 制御構文Bの条件式を表記する</td></tr>
<tr><td>B-05</td><td>補足</td><td></td><td>例えば, 選択された状態変数がSTATEで,
if (EVENT==1){
  if(STATE==1){
  STATE=0;
  } else {
  STATE=1;
  }
}
という構文では, if(EVENT==1){} の構文は, 状態変数を含むif構文 if(STATE==1){} else {} を含むネストに存在する.
したがって, 条件式 EVENT==1 が STATE=0; STATE=1;と同じ行に出力される.
右表は, 上記の構文を対象に作成される状態遷移表である.（"関数"は上記の構文が含まれる関数名を指す）
（赤:制御構文A, 青:制御構文B）
<table><tr><td></td><td></td><td>STATE</td><td></td></tr><tr><td></td><td></td><td>1</td><td>else</td></tr><tr><td>関数</td><td>EVENT==1</td><td>STATE=0;</td><td>STATE=1;</td></tr></table></td></tr>
<tr><td>B-05-01</td><td>要求</td><td>高</td><td>B-05における制御構文Aのコードブロックの中にif構文, switch構文が存在する場合, その構文は処理の一部として扱う</td></tr>
<tr><td>B-05-01</td><td>補足</td><td></td><td>例えば, 選択された状態変数がSTATEで,
if (EVENT==1){
  if(STATE==1){
  STATE=0;
  if (a==0) {
  proc();
  }
  } else {
  STATE=1;
  }
}
という構文では, if (a==0){ proc(); } の構文はif(STATE==1){}のコードブロックの中にあるため, 処理として扱う.
右表は, 上記の構文を対象に作成される状態遷移表である.（"関数"は上記の構文が含まれる関数名を指す）
（赤:制御構文A, 青:制御構文B）
<table><tr><td></td><td></td><td>STATE</td><td></td></tr><tr><td></td><td></td><td>1</td><td>else</td></tr><tr><td>関数</td><td>EVENT==1</td><td>STATE=0;
if(a==0){
proc();
}</td><td>STATE=1;</td></tr></table></td></tr>
<tr><td>B-05-01-01</td><td>要求</td><td>高</td><td>あるイベント・状態における処理・状態変数の更新は, if構文のブロックごとに状態遷移表の1つのセルに表記する</td></tr>
<tr><td>B-05-01-01</td><td>補足</td><td></td><td>例えば, 選択された状態変数がSTATEで,
if (EVENT==1){
  if(STATE==1){
  if (a==0) {
  proc();
  }
  } else {
  STATE=1;
  }
}
という構文の場合, 右表1が作成される（"関数"は上記の構文が含まれる関数名を指す）.
右表2のようにif構文のソースコード1行ごとに1つのセルが作成されることはない.
表1
<table><tr><td></td><td></td><td>STATE</td><td></td></tr><tr><td></td><td></td><td>1</td><td>else</td></tr><tr><td>関数</td><td>EVENT==1</td><td>if(a==0){
proc();
}</td><td>STATE=1;</td></tr></table>
表2
<table><tr><td></td><td></td><td>STATE</td><td></td></tr><tr><td></td><td></td><td>1</td><td>else</td></tr><tr><td>関数</td><td>EVENT==1</td><td>if(a==0){</td><td>STATE=1;</td></tr><tr><td></td><td></td><td>proc();</td><td></td></tr><tr><td></td><td></td><td>}</td><td></td></tr></table></td></tr>
<tr><td>B-05-01-02</td><td>要求</td><td>高</td><td>if構文のブロック内に状態変数の更新がある場合, 状態変数の更新の行のみ赤色で着色する</td></tr>
<tr><td>B-05-01-02</td><td>補足</td><td></td><td>例えば, 選択された状態変数がSTATEで,
if (EVENT==1){
  if(STATE==1){
  if (a==0) {
  STATE=2;
  }
  } else {
  STATE=1;
  }
}
という構文の場合, 右表が作成される（"関数"は上記の構文が含まれる関数名を指す）.
<table><tr><td></td><td></td><td>STATE</td><td></td></tr><tr><td></td><td></td><td>1</td><td>else</td></tr><tr><td>関数</td><td>EVENT==1</td><td>if(a==0){
STATE=2;
}</td><td>STATE=1;</td></tr></table></td></tr>
</table>

<table>
<tr><td>B-06</td><td>要求</td><td>高</td><td>B-05における制御構文Aを含むネストの外側にある処理・条件式は，状態遷移表に表記しない
ただし，ネスト外にある無条件処理は，以下の場合，削除の対象外とする
1) その処理に状態遷移を含む 2)その処理のネストに他のイベントが存在しない</td></tr>
<tr><td>B-06</td><td>補足</td><td></td><td>例えば，選択された状態変数がSTATEで，
if(a==0){
proc();
if(STATE==1){
STATE=0;
}else{
STATE=1;
}
}
if(b==0){
proc2();
} の構文では，if(b==0){proc2();} は状態遷移表に表記しない．
右表は，上記の構文を対象に作成される状態遷移表である．（"関数"は上記の構文が含まれる関数名を指す）
<table><tr><td></td><td></td><td>STATE</td><td></td></tr><tr><td></td><td></td><td>1</td><td>else</td></tr><tr><td>関数</td><td>a==0</td><td>proc();</td><td>proc();</td></tr><tr><td></td><td></td><td>STATE=0;</td><td>STATE=1;</td></tr></table></td></tr>
<tr><td>B-06</td><td>補足</td><td></td><td>選択された状態変数がSTATE(グローバル変数)で，
void hoge(fuga){
STATE=1;
if(EVENT==0){
if(STATE==1){
STATE=0;
} else {
STATE=1;
}
}
}
void hoge2(fuga2){
STATE=1;
}
という2つの関数がある場合，
関数hogeは無条件実行されるSTATE=1;の処理含め，すべての処理が状態遷移表に表記される．
関数hoge2については状態遷移表を作成しない．（赤：制御構文A，青：制御構文B）</td></tr>
<tr><td>B-07</td><td>要求</td><td>高</td><td>ある状態変数に関する状態遷移表において，すべての処理・状態変数の更新にイベントが存在しない場合，
すべての処理・状態変数の更新に「無条件」というイベントを1列分追加する</td></tr>
</table>

<table>
<tr><td>B-08</td><td>要求</td><td>高</td><td>B-05における制御構文Aに else文のないif構文が存在する場合，該当する状態遷移表の状態にelseを追加する
このとき，制御構文Aの外側で実行される処理・状態変数の更新があれば，状態elseの列に表記する</td></tr>
<tr><td>B-08</td><td>補足</td><td></td><td>例えば，選択された状態変数がSTATEで，
if (a==0) {
if (STATE ==1) {
STATE=0;
}
proc1();
}
<table><tr><td></td><td></td><td>STATE</td><td></td></tr><tr><td></td><td></td><td>1</td><td>else</td></tr><tr><td>関数</td><td>a==0</td><td>STATE=0;</td><td>proc1();</td></tr><tr><td></td><td></td><td>proc1();</td><td></td></tr></table>
という構文の場合，if(STATE==1) のブロックの後にelse文が存在するものとして，状態遷移表に状態elseを追加する．
右表は，上記の構文を対象に作成される状態遷移表である．(″関数″は上記の構文が含まれる関数名を指す)
また，if(STATE==1) を含む制御構文の外側で実行される proc1(); はelse列にも表記される．
(赤:制御構文A，青:制御構文B)</td></tr>
<tr><td>B-09</td><td>要求</td><td>高</td><td>ある関数において，B-05における制御構文Aが複数存在する場合，それらの制御構文ごとに状態遷移表を作成する</td></tr>
<tr><td>B-09</td><td>補足</td><td></td><td>例えば，選択された状態変数がSTATEで，
if (a==0) {
if (STATE ==1) {
STATE=0;
} else {
proc1();
}
}
if (b==0) {
if (STATE==0) {
STATE=0;
proc2();
} else {
proc3();
}
}
<table><tr><td></td><td></td><td>STATE</td><td></td></tr><tr><td></td><td></td><td>1</td><td>else</td></tr><tr><td>関数</td><td>a==0</td><td>STATE=0;</td><td>proc1();</td></tr></table>
<table><tr><td></td><td></td><td>STATE</td><td></td></tr><tr><td></td><td></td><td>0</td><td>else</td></tr><tr><td>関数</td><td>b==0</td><td>STATE=0;</td><td>proc3();</td></tr><tr><td></td><td></td><td>proc2();</td><td></td></tr></table>
の場合，右表が作成される(″関数″は上記の構文が含まれる関数名を指す)．
(赤:制御構文A，青:制御構文B)</td></tr>
<tr><td>B-10</td><td>要求</td><td>中</td><td>状態遷移表作成の内部処理が，操作開始後60秒以内で終了しない場合は，60秒が経過した時点で処理を中止してダイアログを表示する</td></tr>
<tr><td>B-10</td><td>補足</td><td></td><td>内部処理後，VBAによって表が作成されるまでの時間は60秒を超えてもよい．</td></tr>
<tr><td colspan="4">C動作環境</td></tr>
<tr><td>C-01</td><td>要求</td><td>高</td><td>windows10(64bit)，windows7(32bit)でA,Bの要求をすべて網羅できる</td></tr>
<tr><td>C-02</td><td>要求</td><td>中</td><td>windows10(32bit)，windows7(64bit)でA,Bの要求をすべて網羅できる</td></tr>
<tr><td>C-03</td><td>要求</td><td>低</td><td>windows8.1(32bit/64bit)でA,Bの要求をすべて網羅できる</td></tr>
<tr><td>C-04</td><td>要求</td><td>低</td><td>windows10，7，8.1(32bit/64bit)以外のOSで条件処理表，状態遷移表作成の処理が実行された場合は，
警告のダイアログを表示して，出力される結果が正しいことを保証できないことを知らせる</td></tr>
<tr><td>C-05</td><td>要求</td><td>中</td><td>Office2010，2013，2016でA,Bの要求をすべて網羅できる</td></tr>
<tr><td>C-06</td><td>要求</td><td>低</td><td>Office2010，2013，2016以外のMSOfficeソフトウエアで条件処理表，状態遷移表作成の処理が実行された場合は，警告ダイアログを表示する</td></tr>
</table>