CSE 244 Final Project

Haibei Zhang

Dec 17, 2002

Includes:

Haibei_proj3.lex
Haibei_proj3.y
Sample.txt
Output.c
Sample1.txt
Output.c
Sample2.txt
Output.c
Sample3.txt
Output.c
Sample4.txt
Output.c
Sample5.txt
Output.c
Sample6.txt
Output.c
Console screen shot

```
%{
#include <stdlib.h>
#include <string.h>
entry *install_id(char *);
%}

id [A-Za-z][A-Za-z0-9]*
num [0-9]+
ws [ \t\n]+

%%

{ws}            /*doing nothing*/
"begin"         return MYBEGIN;
"end"           return END;
"input"         return INPUT;
"output"        return OUTPUT;
"while"         return WHILE;
"do"            return DO;
"function"      return FUNCTION;
"main"          return MAIN;
"var"           return VAR;
">="            return GREATER_OR_EQUAL;
"<="            return LESS_OR_EQUAL;
"!="            return NOT_EQUAL;
">"                     return GREATER;
"<"                     return LESS;
"="                     return ('=');
"+"                     return ('+');
"-"                     return ('-');
"*"                     return ('*');
"("                     return ('(');
")"                     return (')');
";"                     return (';');
","                     return (',');
{id}            {
                                temp_n=tableLen;
                                yylval.id_type=install_id(yytext);
                                return ID;
            }
{num}           {
                                yylval.num_type=atoi(yytext);
                                return NUM;
            }
.                       return(yytext[0]);

%%

entry *install_id(char *symbol_name)
        {
                entry *ptr=table;
                for(i=0;i<=tableLen;i++)
                        {
                                if(strcmp(symbol_name, table[i].name)==0)
                                {
                                        ptr=&table[i];
                                        return ptr;
                                }
                        }
                tableLen++;
                strcpy(table[tableLen].name,symbol_name);
                ptr=&table[tableLen];
                return ptr;
        }

```

```
%{
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_ID_LEN 100
#define MAX_TABLE_LEN 100
#define MAX_PARAM_NUMBER 100
#define MAX_VAR_NUMBER 100
#define MAX_STMT_LEN 1000
#define MAX_STMT_NUMBER 100
#define MAX_NODE_NUMBER 1000
typedef struct                                          /*The temporary symbol table
 {
        char name[MAX_ID_LEN];
        int isDeclared;
        int isInitiated;
 }entry;

typedef struct                                          /*The main symbol table stru
 {
        char name[MAX_ID_LEN];
        int isDeclared;
        int paramNum;
        struct
        {
                char paramName[MAX_ID_LEN];
                int paramVal;
        }param[MAX_PARAM_NUMBER];
        int varNum;
        struct
        {
                char varName[MAX_ID_LEN];
                int varVal;
        }var[MAX_VAR_NUMBER];
        int statemtNum;
        char statemt[MAX_STMT_NUMBER][MAX_STMT_LEN];
        char output[MAX_ID_LEN];
 }funcentry;

typedef struct                                          /*The node structure*/
 {
        int funcVal;                                    /*Synthesized, to count actu
        char stmtVal[MAX_STMT_LEN];                     /*Synthesized, Content of statement,
 }node;


int clearTable();
int checkID(int, char *);
int checkFunc(char *, int);
int addFunc(char *);
int addParam(int, char *);
int addVar(int, char *);
int addStmt(int, char *);
node *initNode();

char mainFunction[MAX_ID_LEN]="undefined";              /*name of main function decl
int finalMainFunction=-1;                                       /*index of f
int tableLen=0;                                                 /*ac
int functableLen=0;                                             /*ac
int currentfunc=0;                                              /*cu
int temp_n=0;
char tempstring[MAX_ID_LEN];
node tempnode[MAX_NODE_NUMBER];
int nodeNum=0;
int i=0;
int j=0;
FILE *fo;
entry table[MAX_TABLE_LEN];                             /*The temporary symbol table used by
funcentry functable[MAX_TABLE_LEN];             /*The main symbol table*/
%}


%union{
                int num_type;
                entry *id_type;
```

```
77                  node *node_type;
78                  }
79
80  %token MYBEGIN
81  %token END
82  %token INPUT
83  %token OUTPUT
84  %token WHILE
85  %token DO
86  %token FUNCTION
87  %token MAIN
88  %token VAR
89  %token GREATER_OR_EQUAL
90  %token LESS_OR_EQUAL
91  %token GREATER
92  %token LESS
93  %token NOT_EQUAL
94  %token <num_type> NUM
95  %token <id_type> ID
96  %type <node_type> function
97  %type <node_type> func_body
98  %type <node_type> para
99  %type <node_type> para_list
100 %type <node_type> var_list
101 %type <node_type> stmt_list
102 %type <node_type> stmt
103 %type <node_type> asmt
104 %type <node_type> whil
105 %type <node_type> expr
106 %type <node_type> func_call
107 %type <node_type> actual_para
108 %type <node_type> actual_para_list
109 %type <node_type> relat
110 %type <node_type> whilstmt_list
111
112 %left '+' '-' '*'
113 %right '='
114
115 %%
116 prog            :
117                         MAIN FUNCTION '=' ID ';'
118                         {
119                                 strcpy(mainFunction, $4->name);
120                                 clearTable();
121                         }
122                         func_list
123                 ;
124
125 func_list       :
126                         function
127                         {
128                                 clearTable();
129                         }
130                         func_list
131                 |       /*e*/
132                 ;
133
134 function        :
135                         FUNCTION ID
136                         {
137                                 $$=initNode();
138                                 if(addFunc($2->name)==1)
139                                         {
140                                                 currentfunc=functableLen-1;
141                                         }
142                                 else
143                                         yyerror("Cannot declare function, function name alre
144                                 yyerrok;
145                         }
146                         '('     INPUT para ')' MYBEGIN func_body OUTPUT ID ';'
147                         {
148                                 if(checkID(currentfunc, $11->name)>0)
149                                         strcpy(functable[currentfunc].output, $11->name);
150                                 else
151                                 {
152                                         yyerror("Output not declared");
```

```
153                                     strcpy(functable[currentfunc].output, "0/*output not
154                             }
155                             yyerrok;
156                     }
157                     END
158             ;
159
160 para            :
161                     para_list
162             |       /*e*/
163                     {
164                             functable[currentfunc].paramNum=0;
165                     }
166             ;
167
168 para_list       :
169                     ID ',' para_list
170                     {
171                             if(addParam(currentfunc, $1->name)==0)
172                                     yyerror("Max number of parameter reached");
173                             yyerrok;
174                     }
175             |       ID
176                     {
177                             if(addParam(currentfunc, $1->name)==0)
178                                     yyerror("Max number of parameter reached");
179                             yyerrok;
180                     }
181             ;
182
183 func_body       :
184                     VAR     var_list ';' stmt_list
185                     {}
186             ;
187
188 var_list        :
189                     ID ',' var_list
190                     {
191                             if(checkID(currentfunc, $1->name)!=1)
192                             {
193                                     if(addVar(currentfunc, $1->name)==0)
194                                             yyerror("Max number of var reached");
195                                     yyerrok;
196                             }
197                             else
198                             {
199                                     yyerror("Var already defined in parameter");
200                                     yyerrok;
201                             }
202                     }
203             |       ID
204                     {
205                             if(checkID(currentfunc, $1->name)!=1)
206                             {
207                                     if(addVar(currentfunc, $1->name)==0)
208                                             yyerror("Max number of var reached");
209                                     yyerrok;
210                             }
211                             else
212                             {
213                                     yyerror("Var already defined in parameter");
214                                     yyerrok;
215                             }
216                     }
217             ;
218
219 stmt_list       :
220                     stmt ';' stmt_list
221                     {
222                             if(addStmt(currentfunc, $1->stmtVal)==0)
223                                     yyerror("Max number of statement reached");
224                             yyerrok;
225                     }
226             |       stmt ';'
227                     {
228                             if(addStmt(currentfunc, $1->stmtVal)==0)
```

```
229                                     yyerror("Max number of statement reached");
230                             yyerrok;
231                     }
232             ;
233
234 stmt            :
235                     asmt
236                     {
237                             $$=initNode();
238                             strcpy($$->stmtVal,$1->stmtVal);
239                     }
240             |       whil
241                     {
242                             $$=initNode();
243                             strcpy($$->stmtVal,$1->stmtVal);
244                     }
245             ;
246
247 asmt            :
248                     ID '=' expr
249                     {
250                             $$=initNode();
251                             if(checkID(currentfunc, $1->name)==0)
252                                     {
253                                             strcpy($$->stmtVal,"/*Undeclared ID used in
254                                             strcpy($$->stmtVal, strcat($$->stmtVal, $1->
255                                             addVar(currentfunc, $1->name);
256                                             yyerror("Undeclared ID used in assignment");
257                                     }
258                             else
259                                     strcpy($$->stmtVal,$1->name);
260                             strcpy($$->stmtVal, strcat($$->stmtVal, " = "));
261                             strcpy($$->stmtVal, strcat($$->stmtVal, $3->stmtVal));
262                     }
263             |       expr
264                     {
265                             $$=initNode();
266                             strcpy($$->stmtVal,"/*Isolated expression: must be evaluated
267                             strcpy($$->stmtVal, strcat($$->stmtVal, "/*"));
268                             strcpy($$->stmtVal, strcat($$->stmtVal, $1->stmtVal));
269                             strcpy($$->stmtVal, strcat($$->stmtVal, "*/"));
270                             yyerror("Isolated expression");
271                             yyerrok;
272                     }
273             |       ID '=' '=' expr
274                     {
275                             $$=initNode();
276                             strcpy($$->stmtVal,"/*Invalid assignment using ==, corrected
277                             strcpy($$->stmtVal, strcat($$->stmtVal, $1->name));
278                             strcpy($$->stmtVal, strcat($$->stmtVal, "="));
279                             strcpy($$->stmtVal, strcat($$->stmtVal, $4->stmtVal));
280                             yyerror("Invalid assignment");
281                             yyerrok;
282                     }
283             ;
284
285 expr            :
286                     expr '+' expr
287                     {
288                             $$=initNode();
289                             strcpy($$->stmtVal,strcat($1->stmtVal, " + "));
290                             strcpy($$->stmtVal,strcat($$->stmtVal, $3->stmtVal));
291                     }
292             |       expr '-' expr
293                     {
294                             $$=initNode();
295                             strcpy($$->stmtVal,strcat($1->stmtVal, " - "));
296                             strcpy($$->stmtVal,strcat($$->stmtVal, $3->stmtVal));
297                     }
298             |       '(' expr ')'
299                     {
300                             $$=initNode();
301                             strcpy($$->stmtVal,strcat("(", $2->stmtVal));
302                             strcpy($$->stmtVal,strcat($$->stmtVal, ")"));
303                     }
304             |       expr '*' expr
```

```
305                         {
306                                 $$=initNode();
307                                 strcpy($$->stmtVal,strcat($1->stmtVal, " * "));
308                                 strcpy($$->stmtVal,strcat($$->stmtVal, $3->stmtVal));
309                         }
310                 |       ID
311                         {
312                                 $$=initNode();
313                                 if(checkID(currentfunc, $1->name)==0)
314                                 {
315                                         strcpy($$->stmtVal,"/*Undeclared ID used in assignme
316                                         strcpy($$->stmtVal, strcat($$->stmtVal, $1->name));
317                                         addVar(currentfunc, $1->name);
318                                         yyerror("Undeclared ID used in expression"); yyerrok
319                                 }
320                                 else
321                                 {
322                                         strcpy($$->stmtVal,$1->name);
323                                 }
324                         }
325                 |       func_call
326                         {
327                                 $$=initNode();
328                                 strcpy($$->stmtVal,$1->stmtVal);
329                         }
330                 |       NUM
331                         {
332                                 $$=initNode();
333                                 strcpy($$->stmtVal,itoa($1, tempstring, 10));
334                         }
335                 ;
336
337 func_call       :
338                         ID '(' actual_para ')'
339                         {
340                                 $$=initNode();
341                                 if(checkFunc($1->name, $3->funcVal)>=0)
342                                         {
343                                                 strcpy($$->stmtVal,strcat($1->name, "("));
344                                                 strcpy($$->stmtVal,strcat($$->stmtVal, $3->s
345                                                 strcpy($$->stmtVal,strcat($$->stmtVal, ")"))
346                                         }
347                                 else
348                                         {
349                                                 strcpy($$->stmtVal, "0/*Undeclared function
350                                                 yyerror("Undeclared function in function cal
351                                                 yyerrok;
352                                         }
353                         }
354                 ;
355
356 actual_para                     :
357                         actual_para_list
358                         {
359                                 $$=initNode();
360                                 $$->funcVal=$1->funcVal;
361                                 strcpy($$->stmtVal,$1->stmtVal);
362                         }
363                 |       /*e*/
364                         {
365                                 $$=initNode();
366                                 $$->funcVal=0;
367                                 strcpy($$->stmtVal, "");
368                         }
369                 ;
370
371 actual_para_list        :
372                         NUM ',' actual_para_list
373                         {
374                                 $$=initNode();
375                                 $$->funcVal=$3->funcVal+1;
376                                 strcpy($$->stmtVal,strcat(itoa($1, tempstring, 10),", "));
377                                 strcpy($$->stmtVal,strcat($$->stmtVal, $3->stmtVal));
378                         }
379                 |       NUM
380                         {
```

```
381                             $$=initNode();
382                             $$->funcVal=1;
383                             strcpy($$->stmtVal, itoa($1, tempstring, 10));
384                     }
385             |       ID ',' actual_para_list
386                     {
387                             $$=initNode();
388                             if(checkID(currentfunc, $1->name)>0)
389                             {
390                                     strcpy($$->stmtVal,$1->name);
391                             }
392                             else
393                             {
394                                     strcpy($$->stmtVal,"/*Undeclared ID used in assignme
395                                     strcpy($$->stmtVal, strcat($$->stmtVal, $1->name));
396                                     addVar(currentfunc, $1->name);
397                                     yyerror("Undeclared ID used in function call"); yyer
398                             }
399                             strcpy($$->stmtVal,strcat($$->stmtVal, ", "));
400                             strcpy($$->stmtVal,strcat($$->stmtVal, $3->stmtVal));
401                             $$->funcVal=$3->funcVal+1;
402                     }
403             |       ID
404                     {
405                             $$=initNode();
406                             if(checkID(currentfunc, $1->name)>0)
407                             {
408                                     strcpy($$->stmtVal,$1->name);
409                             }
410                             else
411                             {
412                                     strcpy($$->stmtVal,"/*Undeclared ID used in assignme
413                                     strcpy($$->stmtVal, strcat($$->stmtVal, $1->name));
414                                     addVar(currentfunc, $1->name);
415                                     yyerror("Undeclared ID used in function call"); yyer
416                             }
417                             $$->funcVal=1;
418                     }
419             ;
420
421 whil            :
422                     WHILE '(' expr relat expr ')' DO MYBEGIN whilstmt_list END
423                     {
424                             $$=initNode();
425                             strcpy($$->stmtVal,"while (");
426                             strcpy($$->stmtVal, strcat($$->stmtVal,$3->stmtVal));
427                             strcpy($$->stmtVal, strcat($$->stmtVal, $4->stmtVal));
428                             strcpy($$->stmtVal, strcat($$->stmtVal, $5->stmtVal));
429                             strcpy($$->stmtVal, strcat($$->stmtVal, ")\n{\n"));
430                             strcpy($$->stmtVal, strcat($$->stmtVal, $9->stmtVal));
431                             strcpy($$->stmtVal, strcat($$->stmtVal, "}\n"));
432                     }
433             ;
434
435 relat           :
436                     GREATER_OR_EQUAL        {$$=initNode();strcpy($$->stmtVal, " >= ");}
437             |       LESS_OR_EQUAL           {$$=initNode();strcpy($$->stmtVal, " <= ");}
438             |       NOT_EQUAL                       {$$=initNode();strcpy($$->stmtVal, "
439             |       GREATER                         {$$=initNode();strcpy($$->stmtVal, "
440             |       LESS                            {$$=initNode();strcpy($$->stmtVal, "
441             |       '='                                     {$$=initNode();strcpy($$->st
442             ;
443
444 whilstmt_list   :
445                     stmt ';' whilstmt_list
446                     {
447                             $$=initNode();
448                             strcpy($$->stmtVal, strcat($1->stmtVal, ";\n"));
449                             strcpy($$->stmtVal, strcat($1->stmtVal, $3->stmtVal));
450
451                     }
452             |       stmt ';'
453                     {
454                             $$=initNode();
455                             strcpy($$->stmtVal, $1->stmtVal);
456                             strcpy($$->stmtVal, strcat($1->stmtVal, ";\n"));
```

```
457                         }
458                 ;
459
460 %%
461 #include "lex.yy.c"
462
463 int clearTable()
464 {
465         for(i=0;i<MAX_TABLE_LEN;i++)
466                 {
467                         strcpy(table[i].name, "");
468                 }
469         tableLen=0;
470         return 1;
471 }
472
473 int checkID(int funcindex, char *id)
474 {
475         for(i=0;i<functable[funcindex].paramNum;i++)
476                 if(strcmp(functable[funcindex].param[i].paramName, id)==0)
477                         return 1;
478         for(i=0;i<functable[funcindex].varNum;i++)
479                 if(strcmp(functable[funcindex].var[i].varName, id)==0)
480                         return 2;
481         return 0;
482 }
483
484 int checkFunc(char *id, int pNum)
485 {
486         for(i=0;i<functableLen;i++)
487                 if((strcmp(functable[i].name, id)==0)&&(functable[i].paramNum==pNum))
488                         return i;
489         return -1;
490 }
491
492 int addFunc(char *newfunc)
493 {
494         for(i=0;i<functableLen;i++)
495                 if(strcmp(functable[i].name,newfunc)==0)
496                         return 0;
497         strcpy(functable[functableLen].name,newfunc);
498         functable[functableLen].isDeclared=1;
499         clearTable();
500         functableLen++;
501         return 1;
502 }
503
504 int addParam(int funcindex, char *id)
505 {
506         int n=functable[funcindex].paramNum;
507         if(n<MAX_PARAM_NUMBER)
508                 {
509                         strcpy(functable[funcindex].param[n].paramName,id);
510                         (functable[funcindex].paramNum)++;
511                         return 1;
512                 }
513         else return 0;
514 }
515
516 int addVar(int funcindex, char *id)
517 {
518         int n=functable[funcindex].varNum;
519         if(n<MAX_VAR_NUMBER)
520                 {
521                         strcpy(functable[funcindex].var[n].varName,id);
522                         (functable[funcindex].varNum)++;
523                         return 1;
524                 }
525         else return 0;
526 }
527
528 int addStmt(int funcindex, char *stmtContent)
529 {
530         int n=functable[funcindex].statemtNum;
531         if(n<MAX_STMT_NUMBER)
532                 {
```

```
533                         strcpy(functable[funcindex].statemt[n], stmtContent);
534                         (functable[funcindex].statemtNum)++;
535                         return 1;
536                 }
537         else return 0;
538 }
539
540 node *initNode()
541 {
542         nodeNum++;
543         return &tempnode[nodeNum];
544 }
545
546 yyerror(s)
547 char *s;
548 {
549   printf ("/*%s\n\t\t*/", s);
550 }
551
552 main(argc, argv )
553 int argc;
554 char **argv;
555 {
556     ++argv, --argc;  /* skip over program name */
557         if ( argc > 0 )
558             yyin = fopen( argv[0], "r" );
559     else
560             yyin = stdin;
561         yyparse();
562
563         fo=fopen("output.c", "w");
564         fprintf(fo, "/*Output C file translated by YACC\n");
565         fprintf(fo, " *CSE 244 Final Project\n");
566         fprintf(fo, " *Haibei Zhang Dec 17, 2002\n");
567         fprintf(fo, " */\n\n");
568         fprintf(fo, "#include<stdlib.h>\n");
569         fprintf(fo, "#include<stdio.h>\n");
570
571         /*Print Functions*/
572         for(i=0;i<functableLen;i++)
573         {
574                 /*Print function declaration*/
575                 fprintf(fo, "int %s(", functable[i].name);
576                 if(functable[i].paramNum>0)
577                 {
578                         for(j=functable[i].paramNum-1;j>0;j--)
579                                 fprintf(fo, "int %s, ", functable[i].param[j].paramName);
580                         fprintf(fo, "int %s", functable[i].param[0].paramName);
581                 }
582                 fprintf(fo, ")\n");
583
584                 /*Print leading brachet*/
585                 fprintf(fo, "{\n");
586
587                 /*Print declaration*/
588                 fprintf(fo, "int ");
589                 if(functable[i].varNum>0)
590                 {
591                         for(j=functable[i].varNum-1;j>0;j--)
592                                 fprintf(fo, "%s, ", functable[i].var[j].varName);
593                         fprintf(fo, "%s", functable[i].var[0].varName);
594                         fprintf(fo, ";\n\n");
595                 }
596
597                 /*Print Statements*/
598                 if(functable[i].statemtNum>0)
599                 {
600                         for(j=functable[i].statemtNum-1;j>0;j--)
601                                 fprintf(fo, "%s; \n", functable[i].statemt[j]);
602                         fprintf(fo, "%s; \n", functable[i].statemt[0]);
603                 }
604
605                 /*Print return value*/
606                 fprintf(fo, "return %s;\n", functable[i].output);
607
608                 /*Print ending bracket*/
```

```
609                 fprintf(fo, "}\n\n\n");
610
611                 if(strcmp(functable[i].name, mainFunction)==0)
612                         finalMainFunction=i;
613         }
614
615         /*Print Main Function*/
616         fprintf(fo, "main(argc, argv )\n");
617         fprintf(fo, "int argc; char **argv;\n");
618         fprintf(fo, "{\n");
619         fprintf(fo, "++argv, --argc;  /* skip over program name */\n");
620         if(finalMainFunction>=0)
621                 {
622                         fprintf(fo, "if ( argc == %d )  /* match argument count with main fu
623                         fprintf(fo, "\tprintf(\"%c%c\\n\", %s(atoi(", '%','d', functable[fin
624                         if(functable[finalMainFunction].paramNum>0)
625                         {
626                                 for(j=0;j<functable[finalMainFunction].paramNum-1;j++)
627                                         fprintf(fo, "argv[%d], ",j);
628                                 fprintf(fo, "argv[%d]",functable[finalMainFunction].paramNum
629                         }
630                         fprintf(fo, ")));\n");
631                         fprintf(fo, "else\n");
632                         fprintf(fo, "\tprintf(\"Input Error: argument count does not match t
633                 }
634         else
635                 fprintf(fo, "/*Error: main function %s not declared*/\n", mainFunction);
636         fprintf(fo, "}");
637
638 }
```

```
main function=k12;

function tau(input x,y)
begin
 var t;
 x=x-2;
 t=1;
 y=y*x+13-t;
 while (x<=10) do
 begin
  t=t+12;
  x=x+1;
 end;
 output t;
end

function k12(input x)
begin
 var t;

 t=tau(x,x);
 while (x<=5) do
 begin
  x=x+1;
  t=t*2;
 end;
 output t;
end
```

```c
/*Output C file translated by YACC
 *CSE 244 Final Project
 *Haibei Zhang Dec 17, 2002
 */

#include<stdlib.h>
#include<stdio.h>
int tau(int x, int y)
{
int t;

x = x - 2;
t = 1;
y = y * x + 13 - t;
while (x <= 10)
{
t = t + 12;
x = x + 1;
}
;
return t;
}


int k12(int x)
{
int t;

t = tau(x, x);
while (x <= 5)
{
x = x + 1;
t = t * 2;
}
;
return t;
}


main(argc, argv )
int argc; char **argv;
{
++argv, --argc;  /* skip over program name */
if ( argc == 1 )  /* match argument count with main function */
    printf("%d\n", k12(atoi(argv[0])));
else
    printf("Input Error: argument count does not match to main function k12\n");
}
```

```
main function=a2;

function a1(input x)
begin
        var i,y;

        i = 1;
        y = 1;

        while (i<=x) do
        begin
                y=y*2;
                i=i+1;
        end;

        output y;
end

function a2(input x)
begin
        var rrr;

        rrr=a1(x);

        output rrr;
end
```

```c
/*Output C file translated by YACC
 *CSE 244 Final Project
 *Haibei Zhang Dec 17, 2002
 */

#include<stdlib.h>
#include<stdio.h>
int a1(int x)
{
int i, y;

i = 1;
y = 1;
while (i <= x)
{
y = y * 2;
i = i + 1;
}
;
return y;
}


int a2(int x)
{
int rrr;

rrr = a1(x);
return rrr;
}


main(argc, argv )
int argc; char **argv;
{
++argv, --argc;  /* skip over program name */
if ( argc == 1 )  /* match argument count with main function */
    printf("%d\n", a2(atoi(argv[0])));
else
    printf("Input Error: argument count does not match to main function a2\n");
}
```

```
main function=doit;

function exponentiate(input x,y)
begin
        var counter, result;

        counter = 1;
        result = 1;

        while (counter<=y) do
        begin
                result = result * x;
                counter = counter + 1;
        end;

        output result;
end

function doit(input x,y,z)
begin
        var result;

        result = x + exponentiate(y,x) + exponentiate(z,x);

        output result;
end

```

```c
/*Output C file translated by YACC
 *CSE 244 Final Project
 *Haibei Zhang Dec 17, 2002
 */

#include<stdlib.h>
#include<stdio.h>
int exponentiate(int x, int y)
{
int counter, result;

counter = 1;
result = 1;
while (counter <= y)
{
result = result * x;
counter = counter + 1;
}
;
return result;
}


int doit(int x, int y, int z)
{
int result;

result = x + exponentiate(y, x) + exponentiate(z, x);
return result;
}


main(argc, argv )
int argc; char **argv;
{
++argv, --argc;  /* skip over program name */
if ( argc == 3 )  /* match argument count with main function */
    printf("%d\n", doit(atoi(argv[0], argv[1], argv[2])));
else
    printf("Input Error: argument count does not match to main function doit\n");
}
```

```
main function=fff;

function fib(input x)
begin
        var temparg,temp1,temp2,result,control;

        control = x;

        while (control=0) do
        begin
                result = 0;
                control = 1;
        end;

        control = x;
        while (control=1) do
        begin
                result = 1;
                control = 0;
        end;

        control = x;
        while (control>1) do
        begin
                temparg = x-1;
                temp1 = fib(temparg);
                temparg = x-2;
                temp2 = fib(temparg);
                result = temp1 + temp2;
                control = 0;
        end;

        output result;
end

function nvfib(input x)
begin
        var cond,argg,temp;

        argg = 0;
        cond = fib(argg);

        while( cond <= x) do
        begin
                argg=argg+1;
                cond = fib(argg);
        end;

        argg = argg - 1;

        output argg;
end

function fff(input x)
begin
        var result;

        result = nvfib(x);

        output result;
end

```

```c
/*Output C file translated by YACC
 *CSE 244 Final Project
 *Haibei Zhang Dec 17, 2002
 */

#include<stdlib.h>
#include<stdio.h>
int fib(int x)
{
int temparg, temp1, temp2, result, control;

control = x;
while (control == 0)
{
result = 0;
control = 1;
}
;
control = x;
while (control == 1)
{
result = 1;
control = 0;
}
;
control = x;
while (control > 1)
{
temparg = x - 1;
temp1 = fib(temparg);
temparg = x - 2;
temp2 = fib(temparg);
result = temp1 + temp2;
control = 0;
}
;
return result;
}


int nvfib(int x)
{
int cond, argg, temp;

argg = 0;
cond = fib(argg);
while (cond <= x)
{
argg = argg + 1;
cond = fib(argg);
}
;
argg = argg - 1;
return argg;
}


int fff(int x)
{
int result;

result = nvfib(x);
return result;
}


main(argc, argv )
int argc; char **argv;
{
++argv, --argc;  /* skip over program name */
if ( argc == 1 )  /* match argument count with main function */
    printf("%d\n", fff(atoi(argv[0])));
else
    printf("Input Error: argument count does not match to main function fff\n");
}
```

```
main function=a3;

function a1(input x,y)
begin
        var i;

        i = 1;
        myvar = 2;

        while (i<=x) do
        begin
                y=y*i;
                i=i+1;
        end;

        output myvar;
end

function a2(input x)
begin
        var rrr,temp;

        temp = x-1;

        rrr=a1(x, x) + a1(temp);

        output rrr;
end

function  a3(input empty)
begin
        var a;

        a = 0;
        a = a2(a);

        output a;
end
```

```c
/*Output C file translated by YACC
 *CSE 244 Final Project
 *Haibei Zhang Dec 17, 2002
 */

#include<stdlib.h>
#include<stdio.h>
int a1(int x, int y)
{
int myvar, i;

i = 1;
/*Undeclared ID used in assignment, declaration added by parser*/
myvar = 2;
while (i <= x)
{
y = y * i;
i = i + 1;
}
;
return myvar;
}


int a2(int x)
{
int rrr, temp;

temp = x - 1;
rrr = a1(x, x) + 0/*Undeclared function in function call*/;
return rrr;
}


int a3(int empty)
{
int a;

a = 0;
a = a2(a);
return a;
}


main(argc, argv )
int argc; char **argv;
{
++argv, --argc;  /* skip over program name */
if ( argc == 1 )  /* match argument count with main function */
    printf("%d\n", a3(atoi(argv[0])));
else
    printf("Input Error: argument count does not match to main function a3\n");
}
```

```
main function=a3;

function a1(input x)
begin
        var x,myvar,a2,a5;

        x = 1;
        myvar = 2;

        while (i<=x) do
        begin
                y=y*i;
                i=i+1;
        end;

        output myvar;
end

function a2(input x)
begin
        var rrr,temp;

        temp = x-1;
        temp2 = 3;

        rrr=a1(x) + a1(temp);

        output rrr;
end

function  a3(input empty)
begin
        var a;

        a = 0;
        a2(a);

        output a;
end
```

```c
/*Output C file translated by YACC
 *CSE 244 Final Project
 *Haibei Zhang Dec 17, 2002
 */

#include<stdlib.h>
#include<stdio.h>
int a1(int x)
{
int y, i, myvar, a2, a5;

x = 1;
myvar = 2;
while (/*Undeclared ID used in assignment, declared by parser*/i <= x)
{
y = /*Undeclared ID used in assignment, declared by parser*/y * i;
i = i + 1;
}
;
return myvar;
}


int a2(int x)
{
int temp2, rrr, temp;

temp = x - 1;
/*Undeclared ID used in assignment, declaration added by parser*/
temp2 = 3;
rrr = a1(x) + a1(temp);
return rrr;
}


int a3(int empty)
{
int a;

a = 0;
/*Isolated expression: must be evaluated in a statement*/
/*a2(a)*/;
return a;
}


main(argc, argv )
int argc; char **argv;
{
++argv, --argc;  /* skip over program name */
if ( argc == 1 )  /* match argument count with main function */
    printf("%d\n", a3(atoi(argv[0])));
else
    printf("Input Error: argument count does not match to main function a3\n");
}
```

```
main function=rrr;

function a1(input x)
begin
        var x;

        x = 1;
        myvar = 2;

        while (i<=x) do
        begin
                y==y*2;
                i=i+1;
        end;

        output y;
end

function a2(input x)
begin
        var rrr,temp,temp2;

        temp = x-1;
        temp2 = 3;

        rrr=a1(x) + a1(0);

        output rrr;
end

function  a3(input x,y,z)
begin
        var b;

        a = 0;
        a = a2(a);

        output a;
end
```

```c
/*Output C file translated by YACC
 *CSE 244 Final Project
 *Haibei Zhang Dec 17, 2002
 */

#include<stdlib.h>
#include<stdio.h>
int a1(int x)
{
int y, i, myvar;

x = 1;
/*Undeclared ID used in assignment, declaration added by parser*/
myvar = 2;
while (/*Undeclared ID used in assignment, declared by parser*/i <= x)
{
/*Invalid assignment using ==, corrected by parser*/
y=/*Undeclared ID used in assignment, declared by parser*/y * 2;
i = i + 1;
}
;
return y;
}


int a2(int x)
{
int rrr, temp, temp2;

temp = x - 1;
temp2 = 3;
rrr = a1(x) + a1(0);
return rrr;
}


int a3(int x, int y, int z)
{
int a, b;

/*Undeclared ID used in assignment, declaration added by parser*/
a = 0;
a = a2(a);
return a;
}


main(argc, argv )
int argc; char **argv;
{
++argv, --argc;  /* skip over program name */
/*Error: main function rrr not declared*/
}
```

Console screen shot: