****

**Ahsanullah University of Science and Technology**

**Department of Computer Science and Engineering**

**Course No. :** CSE4130

**Course Name :** Formal Languages and

Compilers Lab

**Final Project**

**Submitted By:**

Name : Shahriar Hasan Chowdhury

ID No. : 17 01 04 030

Session : Spring - 2020

Section : A (A2)

**Assignment Question:**

Write a C program to read a source code in C from the console and print a report on the given

source code. The report will contain the following:

1. The source code with line number

2. Any kind of error covered in Session 1 to Session 5

3. Header Files

4. Identifiers

5. Keywords

6. Functions

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| #include<stdio.h>  /\*Program Starts\*/  float func(int x))  {  int y;;  for(y; x, ){} //for loop  return 1.5;  }  int main()  {  int a=1;  if(c)  a=2;  else  a=0;  return 0;  } | Source code with line number:  1. #include<stdio.h>  2. /\*Program Starts\*/  3. float func(int x))  4. {  5. int y;;  6. for(y; x, ){} //for loop  7. return 1.5;  8. ..........  9. ..........  Errors:  At line number 3: Misplaced Parenthesis  At line number 5: ..............................  At line number 6: ..............................  ......................................................  Header Files: stdio.h  Identifiers: func, x, y, ............  Keywords: float, int, .............  Functions: func, main |

**Code:**

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\* CSE 4130 : Spring 2020 Final Project ID: 170104030 \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include<stdio.h>  #include<string.h>  #include<ctype.h>  int max\_word = 10000;  int each\_word\_len = 20;  void categorizeLexemes(char input2[]);  int isLine(char arr[]);  // ============================================ assignment 1 starts ================================================  FILE \*p1,\*p2, \*p3,\*output\_file;  void add\_line()  {      FILE \*f1 = fopen("without\_comment.txt", "r");      FILE \*f2 = fopen("with\_line.txt", "w");      int line\_num = 1;      char line[20];      itoa(line\_num, line, 10);      fputs("line ", f2);      line\_num++;      char c;      if(!f1)      {          printf("\nFile not found");      }      else      {          while((c = fgetc(f1)) != EOF)          {              if((c == '\n'))              {                  itoa(line\_num, line, 10);                  fputc('\n', f2);                  fputs("line ", f2);                  line\_num++;              }              else              {                  fputc(c,f2);              }          }      }      fclose(f1);      fclose(f2);  }  void print\_source\_code()  {      output\_file = fopen("report\_170104030.txt", "a");      FILE \*f1 = fopen("input.c", "r");      int line\_num = 1;      char line[20];      itoa(line\_num, line, 10);      fprintf(output\_file, "%-3s. ", line);      printf("%-3s. ", line);      line\_num++;      char c;      if(!f1)      {          printf("\ninput file File not found");      }      else      {          while((c = fgetc(f1)) != EOF)          {              if((c == '\n'))              {                  fprintf(output\_file, "\n");                  printf("\n");                  itoa(line\_num, line, 10);                  fprintf(output\_file, "%-3s. ", line);                  printf("%-3s. ", line);                  line\_num++;              }              else              {                  fprintf(output\_file, "%c", c);                  printf("%c", c);              }          }      }      fclose(f1);      fclose(output\_file);  }  void single\_line\_comment()  {      char temp1;      while((temp1=fgetc(p1))!=EOF)      {          if(temp1 == '\n')              return;      }  }  void multi\_line\_comment()  {      char temp1, temp2;      while((temp1=fgetc(p1))!=EOF)      {          if(temp1 == '\*')          {              temp2 = fgetc(p1);              if(temp2=='/')                  return;          }      }  }  void take\_input()  {      char c;      printf("Paste a C code to check: \n\n");      p1 = fopen("input.c", "w");      while(scanf("%c", &c) != EOF){          fputc(c,p1);      }      fclose(p1);  }  void remove\_comments()  {      char c;      p1 = fopen("input.c", "r");      p2 = fopen("without\_comment.txt", "w");      if(!p1)      {          printf("\nFile can't be opened");      }      else      {          while((c=fgetc(p1)) != EOF)          {              if((c =='/'))              {                  if((c=fgetc(p1)) == '/')                  {                      single\_line\_comment();                  }                  else if(c == '\*')                  {                      multi\_line\_comment();                  }              }              else              {                  fputc(c,p2);              }          }      }      fclose(p1);      fclose(p2);  }  void remove\_space\_newline()  {      p1 = fopen("with\_line.txt", "r");      p2 = fopen("output\_assignment\_1.txt", "w");      if(!p1)      {          printf("\nFile can't be opened");      }      else      {          char a,b;          while((a=fgetc(p1)) != EOF)          {              if((a==' '))              {                  char temp = a;                  while((a=fgetc(p1))!=EOF)                  {                      if(a!= ' ' && (a!='\n')&& (a!='\t'))                      {                          fputc(temp,p2);                          break;                      }                  }              }              if((a!=' ') && (a != '\n') && (a!='\t'))              {                  fputc(a,p2);              }          }      }      fclose(p1);      fclose(p2);  }  void call\_assignment1()  {      output\_file = fopen("report\_170104030.txt", "w");      fprintf(output\_file, "Source code with line number:\n\n");      printf("\n\n====================================\n\n");      printf("Source code with line number:\n\n");      fclose(output\_file);      print\_source\_code();      printf("\n");      remove\_comments();      add\_line();      remove\_space\_newline();  }  // --------------------------------------------------- Assignment 1 Ends ------------------------------------  // =================================================== Assignment 2 Starts =====================================  int LINE\_NUM = 1;  int isSeperator(char c)  {      if(c == ';' || c == '\''  || c =='\"' || c == ',')          return 1;      return 0;  }  int isOperator(char c)  {      if(c == '+' || c == '-' || c == '\*' || c == '/' ||          c == '=' || c == '>' || c == '<' || c == '!' ||           c == '%' || c == '&' || c == '^' || c == '~')           return 1;      return 0;  }  int isDoubleOperator(char c1, char c2)  {      if((c1 == '+' && c2 == '+') || (c1 == '-' && c2 == '-') || (c1 == '+' && c2 == '=') ||          (c1 == '-' && c2 == '=') || (c1 == '=' && c2 == '=') || (c1 == '>' && c2 == '=') ||          (c1 == '<' && c2 == '=') || (c1 == '&' && c2 == '&') || (c1 == '|' && c2 == '|') ||          (c1 == '>' && c2 == '>') || (c1 == '<' && c2 == '<') || (c1 == '?' && c2 == ':'))              return 1;      return 0;  }  int isParenthesis(char c)  {      if(c == '(' || c == ')' || c == '{' || c == '}' || c == '[' || c == ']')          return 1;      return 0;  }  char keywords[32][32] =  {      "auto", "const", "double", "float", "int", "short", "struct", "unsigned",      "break", "continue", "else", "for", "long", "signed", "switch", "void",      "case", "default", "enum", "goto", "register", "sizeof", "typedef", "volatile",      "char", "do", "extern", "if", "else", "return", "static", "union", "while"  };    int isKeyword(char arr[])  {      for(int i=0; i<32; i++)      {          if(strcmp(keywords[i], arr) == 0)              return 1;      }      return 0;  }  char line[10][10] = {"line"};  int isLine(char arr[])  {      if(strcmp(line[0], arr) == 0)          return 1;      return 0;  }  int isHeaderInclude(char arr[])  {      if(strcmp(arr, "#include") == 0)          return 1;      return 0;  }  int isIdentifier(char arr[])  {      if (!(isalpha(arr[0]) || arr[0]== '\_'))         return 0;      for (int i = 1; i < strlen(arr); i++)      {          if (!(isalpha(arr[i]) || arr[i] == '\_' || isdigit(arr[i])))              return 0;      }      return 1;  }  int isRealNumber(char arr[])  {      int NumOfPoint = 0;      int digit = 1;      for(int i = 0; i < strlen(arr); i++)      {          if(isdigit(arr[i]))              digit = 1;          else if(arr[i] == '.')              NumOfPoint++;          else          {              digit = 0;              break;          }      }      if(arr[strlen(arr)-1] == '.')          return 0;      if(digit == 1 && NumOfPoint <= 1)          return 1;      return 0;  }  void separateLexemes(char input[])  {      FILE \*f = fopen("step2input.txt", "w");      int l = strlen(input);      for(int i=0;i<l;i++)      {          if(isSeperator(input[i]))          {              fputc(' ', f);              fputc(input[i], f);              fputc(' ', f);          }          else if(isParenthesis(input[i]))          {              fputc(' ', f);              fputc(input[i], f);              fputc(' ', f);          }          else if(isDoubleOperator(input[i], input[i+1]))          {              fputc(' ', f);              fputc(input[i], f);              i++;              fputc(input[i], f);              fputc(' ', f);          }          else if(isOperator(input[i]))          {              fputc(' ', f);              fputc(input[i], f);              fputc(' ', f);          }          else{              fputc(input[i], f);          }      }      fclose(f);  }  void categorizeLexemes(char input2[])  {      FILE \*f2 = fopen("output\_assignment\_2.txt", "w");      int endIndex = 0;      char id[100];      for(int i=0;i<strlen(input2);i++)      {          if(isParenthesis(input2[i]))          {              fputs("[par ", f2);              fputc(input2[i], f2);              fputs("] ", f2);          }          else if(isSeperator(input2[i]))          {              fputs("[sep ", f2);              fputc(input2[i], f2);              fputs("] ", f2);          }          else if(isOperator(input2[i]))          {              if(isOperator(input2[i+1]))              {                  fputs("[op ", f2);                  fputc(input2[i], f2);                  i++;                  fputc(input2[i], f2);                  fputs("] ", f2);              }              else{                  fputs("[op ", f2);                  fputc(input2[i], f2);                  fputs("] ", f2);              }          }          else          {               if(input2[i] != ' ')               {                   id[endIndex] = input2[i];                   endIndex++;               }               else               {                   id[endIndex] = '\0';                   endIndex = 0;                   if(strlen(id) >= 1)                   {                       if(isKeyword(id)){                          fputs("[kw ", f2);                          fputs(id, f2);                          fputs("] ", f2);                       }                       else if(isLine(id)){                          char line[20];                          itoa(LINE\_NUM, line, 10);                          fputs("[line ", f2);                          fputs(line, f2);                          fputs("] ", f2);                          LINE\_NUM++;                       }                       else if(isHeaderInclude(id)){                          fputs("[header ", f2);                          fputs(id, f2);                          fputs("] ", f2);                       }                       else if(isIdentifier(id)){                          fputs("[id ", f2);                          fputs(id, f2);                          fputs("] ", f2);                       }                       else if(isRealNumber(id)){                          fputs("[num ", f2);                          fputs(id, f2);                          fputs("] ", f2);                       }                       else{                          fputs("[unkn ", f2);                          fputs(id, f2);                          fputs("] ", f2);                       }                   }               }          }      }      fclose(f2);  }  void call\_assignment2()  {      char input[100000];      char input2[100000];      FILE \*f1 = fopen("output\_assignment\_1.txt", "r");      if(f1) fgets(input, 100000, f1);      else printf("File Not Found");      fclose(f1);      separateLexemes(input);      FILE \*f2 = fopen("step2input.txt", "r");      if(f2) fgets(input2, 100000, f2);      else printf("File Not Found");      fclose(f2);      categorizeLexemes(input2);  }  /// --------------------------------------------------- Assignment 2 Ends -------------------------------------    /// Making of symble Table #####################################################    int sl = 1;  int totalSymbleInTable = 0;  char maybeAssignmentOP = '!';  int number\_of\_value\_in\_symble\_table;  struct symble\_table  {      int sl\_no;      char name[20];      char id\_type[10];      char data\_type[20];      char scope[10];      char value[100];  };  struct symble\_table st[100];  void keep\_identifier(char input\_step1[],char output\_step1[])  {      int j = 0;      for(int i=0; i<strlen(input\_step1); i++)      {          if(input\_step1[i]=='k' && input\_step1[i+1]=='w')          {              i+=3;          }          else if(input\_step1[i]=='o' && input\_step1[i+1]=='p')          {              i+=3;          }          else if(input\_step1[i]=='n' && input\_step1[i+1]=='u' && input\_step1[i+2]=='m')          {              i+=4;          }          else if(input\_step1[i]=='s' && input\_step1[i+1]=='e' && input\_step1[i+2]=='p')          {              i+=4;          }          else if(input\_step1[i]=='b' && input\_step1[i+1]=='r' && input\_step1[i+2]=='c')          {              i+=4;          }          else if(input\_step1[i]=='p' && input\_step1[i+1]=='a' && input\_step1[i+2]=='r')          {              i+=4;          }          output\_step1[j++]=input\_step1[i];      }  }  int isFunc(char arr[])  {      if(strcmp(arr, "(") == 0)          return 1;      return 0;  }  int k = 1;  void update(int sl\_no, char value[], char isAssign)  {      if(isAssign == '='){          strcpy(st[sl\_no].value, value);      }  }  int checkValidDataTye(char data\_type[])  {      if((strcmp(data\_type, "int") == 0) || (strcmp(data\_type, "float") == 0) || (strcmp(data\_type, "double") == 0) || (strcmp(data\_type, "long") == 0) || (strcmp(data\_type, "short") == 0))          return 1;      else          return 0;  }  int search(char name[], char id\_type[], char scope[])  {      for(int i=0;i<=totalSymbleInTable;i++)      {          if((strcmp(st[i].name, name) == 0) && (strcmp(st[i].scope, scope) == 0)){              return st[i].sl\_no;          }      }      return -1;  }  void insert(int sl\_no, char name[], char id\_type[], char data\_type[], char scope[], char value[], char isAssign)  {      int q = search(name, id\_type, scope);      if(q != -1){          update(q, value, isAssign);      }      else {          if(checkValidDataTye(data\_type)){              st[k].sl\_no = k;              strcpy(st[k].name, name);              strcpy(st[k].id\_type, id\_type);              strcpy(st[k].data\_type, data\_type);              strcpy(st[k].scope, scope);              strcpy(st[k].value, value);              totalSymbleInTable++;              k++;          }      }      number\_of\_value\_in\_symble\_table = k;  }  void display()  {      printf("\t|-------|----------|----------|------------|----------|--------|\n");      printf("\t|%-6s | %-8s | %-8s | %-10s | %-8s | %-6s |\n", "Sl.NO", "Name","Id Type","Data Type","Scope", "Value");      printf("\t|=======|==========|==========|============|==========|========|\n");      for(int i=1; i<k; i++)      {          printf("\t|%-6d | %-8s | %-8s | %-10s | %-8s | %-6s |\n",st[i].sl\_no,st[i].name,st[i].id\_type,st[i].data\_type,st[i].scope,st[i].value);          printf("\t|-------|----------|----------|------------|----------|--------|\n");      }  }  int make\_symble\_table()  {      char input\_step1[100000];      char output\_step1[100000];      char output\_with\_space[100000];  //    FILE \*f1 = fopen("input\_Assignment3.txt", "r");      FILE \*f1 = fopen("output\_assignment\_2.txt", "r");      if(f1)          fgets(input\_step1, 100000, f1);      else          printf("File Not Found");      fclose(f1);      keep\_identifier(input\_step1,output\_step1);      int j = 0;      for(int i=0; i<strlen(output\_step1); i++)      {          if(output\_step1[i] == '[')              continue;          else if(output\_step1[i] == ']')              continue;          else          {              output\_with\_space[j] = output\_step1[i];              j++;          }      }        char word[10000][20];      int k=0;      char\* piece = strtok(output\_with\_space, " ");      while(piece != NULL)      {          strcpy(word[k], piece);          k++;          piece = strtok(NULL, " ");      }      strcpy(word[k], "0--end--0");      int vi = 0;      int si = 0;      char scope[1000][20];      char value[100][20];      char funcOrVar[100][20];      int isScopeGlobal = 1;      strcpy(scope[si],"Global");      // printf("\t===================== Step 2 =====================\n\n");      for(int i=0; i<10000 - 1; i++)      {          if(strcmp("0--end--0", word[i]) == 0)              break;          if(strcmp("id", word[i]) == 0)          {              if(isFunc(word[i+2]))              {                  strcpy(funcOrVar[0],"Func");                  if(checkValidDataTye(word[i-1])){                      insert(sl,word[i+1],funcOrVar[0],word[i-1],scope[si],"\0",maybeAssignmentOP);                  }              }              else              {                  strcpy(funcOrVar[0],"Var");                  if(strcmp("=", word[i+2]) == 0)                  {                      strcpy(value[vi],word[i+3]);                      maybeAssignmentOP = '=';                  }                  insert(sl,word[i+1],funcOrVar[0],word[i-1],scope[si],value[vi],maybeAssignmentOP);                  vi++;                  maybeAssignmentOP = '!';              }              if(strcmp("(", word[i+2]) == 0)              {                  isScopeGlobal = 0;                  strcpy(scope[si],word[i+1]);              }              sl++;          }          if(strcmp("}", word[i]) == 0)          {              isScopeGlobal = 1;              strcpy(scope[si],"Global");          }      }      /// display symble table  //    display();      return 1;  }  /// $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$    void check\_unbalanced\_parenthesis(char word[max\_word][each\_word\_len], int len\_word){      /// check if unbalanced parenthesis      output\_file = fopen("report\_170104030.txt", "a");      int pr1\_start = 0;      int pr1\_end = 0;      int pr2\_start = 0;      int pr2\_end = 0;      int is\_found\_missmatch = 0;      int line\_no = 0;      for(int i=0;i<len\_word;i+=1)      {          if(strcmp(word[i], "line") == 0)          {              line\_no = word[i+1];          }          if(strcmp(word[i], "(") == 0){              pr1\_start++;          }          else if(strcmp(word[i], ")") == 0){              pr1\_end++;          }          else if(strcmp(word[i], "{") == 0){              pr2\_start++;          }          else if(strcmp(word[i], "}") == 0){              pr2\_end++;          }          if(pr1\_end > pr1\_start){              fprintf(output\_file, "At line Number %-3s: Misplaced Parenthesis\n", line\_no);              printf("At line Number %-3s: Misplaced Parenthesis\n", line\_no);              is\_found\_missmatch = 1;              break;          }          if(pr2\_end > pr2\_start){              fprintf(output\_file, "At line Number %-3s: Misplaced Curly Brace\n", line\_no);              printf("At line Number %-3s: Misplaced Curly Brace\n", line\_no);              is\_found\_missmatch = 1;              break;          }      }      if(pr1\_end != pr1\_start && is\_found\_missmatch == 0){          if(pr1\_start > pr1\_end){              fprintf(output\_file, "At line Number %-3s: Misplaced Parenthesis\n", line\_no);              printf("At line Number %-3s: Misplaced Parenthesis\n", line\_no);          }          else{              fprintf(output\_file, "At line Number %-3s: Misplaced Parenthesis\n", line\_no);              printf("At line Number %-3s: Misplaced Parenthesis\n", line\_no);          }      }      if(pr2\_end != pr2\_start && is\_found\_missmatch == 0){          if(pr2\_start > pr2\_end){              fprintf(output\_file, "At line Number %-3s: Misplaced Curly Brace\n", line\_no);              printf("At line Number %-3s: Misplaced Curly Brace\n", line\_no);          }          else{              fprintf(output\_file, "At line Number %-3s: Misplaced Curly Brace\n", line\_no);              printf("At line Number %-3s: Misplaced Curly Brace\n", line\_no);          }      }      fclose(output\_file);      /// end checking unbalanced parenthesis  }  void check\_unbalanced\_else(char word[max\_word][each\_word\_len], int len\_word)  {      /// check unbalanced else      output\_file = fopen("report\_170104030.txt", "a");      int count\_if = 0;      int count\_else = 0;      int is\_found\_missmatch\_else = 0;      int line\_no = 0;      for(int i=0;i<len\_word;i+=1)      {          if(strcmp(word[i], "line") == 0)          {              line\_no = word[i+1];          }          if(strcmp(word[i], "if") == 0){              count\_if++;          }          else if(strcmp(word[i], "else") == 0){              count\_else++;          }          if(count\_else > count\_if){              fprintf(output\_file, "At line Number %-3s: Mismatch Else\n", line\_no);              printf("At line Number %-3s: Mismatch Else\n", line\_no);              break;          }      }      fclose(output\_file);      /// end checking unbalanced else  }  void check\_wrong\_for\_loop(char word[max\_word][each\_word\_len], int len\_word){      output\_file = fopen("report\_170104030.txt", "a");      int line\_no = 0;      for(int i=0;i<len\_word;i+=1)      {          if(strcmp(word[i], "line") == 0)          {              line\_no = word[i+1];          }          if(strcmp(word[i], "for") == 0){              while(strcmp(word[i], ")") != 0)              {                  if(strcmp(word[i], ",") == 0){                      fprintf(output\_file, "At line Number %-3s: Expected ';' before ')'\n", line\_no);                      printf("At line Number %-3s: Expected ';' before ')'\n", line\_no);                  }                  i++;              }          }      }      fclose(output\_file);  }  void check\_undeclared\_id(char word[max\_word][each\_word\_len], int len\_word){      output\_file = fopen("report\_170104030.txt", "a");      int line\_no = 0;      for(int i=0;i<len\_word;i+=1)      {          if(strcmp(word[i], "line") == 0)          {              line\_no = word[i+1];          }          if(strcmp(word[i], "id") == 0){              int temp = number\_of\_value\_in\_symble\_table;              int found = 0;              int j = 0;              for(j=0;j<number\_of\_value\_in\_symble\_table;j++){                  if(strcmp(st[j].name, word[i+1]) == 0){                      found = 1;                      break;                  }              }              if(!found){                  fprintf(output\_file, "At line Number %-3s: Undeclared ID '%s'\n", line\_no,word[i+1]);                  printf("At line Number %-3s: Undeclared ID '%s' \n", line\_no, word[i+1]);              }          }      }      fclose(output\_file);  }  void check\_duplicate\_token(char word[max\_word][each\_word\_len], int len\_word){      output\_file = fopen("report\_170104030.txt", "a");      char crw[20];      int crwi = 0;      char nrw[20];      int nrwi  = 0;      int line\_no = 0;      int err\_code = 1;      int previous\_error\_line = 0;      /// check if duplicate token      for(int i=0;i<len\_word;i+=1)      {          if(strcmp(word[i], "line") == 0)          {              line\_no = word[i+1];          }          strcpy(crw, word[i]);          crwi = i;          strcpy(nrw, word[i+2]);          nrwi = i+2;          if(!(strcmp(crw, "line") == 0) &&             !(strcmp(crw, "par") == 0) &&             !(strcmp(crw, "{") == 0) &&             !(strcmp(crw, "}") == 0) &&             !(strcmp(crw, "(") == 0) &&             !(strcmp(crw, ")") == 0))          {              if(strcmp(crw, nrw) == 0){                  if((strcmp(crw, "kw") == 0) && (strcmp(nrw, "kw") == 0)){                      if((strcmp(word[crwi+1], "else") == 0) && (strcmp(word[nrwi+1], "return") == 0)){                          continue;                      }                  }                  if(previous\_error\_line != line\_no){                      previous\_error\_line = line\_no;                      fprintf(output\_file, "At line Number %-3s: Duplicate Token\n", line\_no);                      printf("At line Number %-3s: Duplicate Token\n", line\_no);                      break;                  }              }          }      }      fclose(output\_file);  }  void check\_error()  {      output\_file = fopen("report\_170104030.txt", "a");      char str[100000];      char output\_step1[100000];      char output\_with\_space[100000];      FILE \*f1 = fopen("output\_assignment\_2.txt", "r");      if(f1)          fgets(str, 100000, f1);      else          printf("File Not Found");      fclose(f1);      /// keep word with space      int j = 0;      for(int i=0; i<strlen(str); i++)      {          if(str[i] == '[')              continue;          else if(str[i] == ']')              continue;          else          {              output\_with\_space[j] = str[i];              j++;          }      }      /// split tokens into words      char word[max\_word][each\_word\_len];      int k=0;      char\* piece = strtok(output\_with\_space, " ");      int len\_word=0;      while(piece != NULL)      {          strcpy(word[k], piece);          k++;          piece = strtok(NULL, " ");          len\_word++;      }      strcpy(word[k], "0--end--0");      /// main work        fprintf(output\_file, "\n\nErrors:\n\n");      printf("\n\nErrors:\n\n");      fclose(output\_file);      check\_unbalanced\_parenthesis(word, len\_word);      check\_unbalanced\_else(word, len\_word);      check\_duplicate\_token(word, len\_word);      check\_wrong\_for\_loop(word, len\_word);      check\_undeclared\_id(word, len\_word);      output\_file = fopen("report\_170104030.txt", "a");      fprintf(output\_file, "\nHeader Files: \t");      printf("\nHeader Files: \t");      for(int i=0;i<len\_word;i++)      {          if(strcmp(word[i], "header") == 0){              fprintf(output\_file,"%s ", word[i+5]);              printf("%s ", word[i+5]);          }      }      fprintf(output\_file,"\nIdentifiers: \t");      printf("\nIdentifiers: \t");      for(int i=0;i<number\_of\_value\_in\_symble\_table;i++)      {          fprintf(output\_file, "%s ",st[i].name);          printf("%s ",st[i].name);      }      fprintf(output\_file,"\nKeyWords: \t");      printf("\nKeyWords: \t");      for(int i=0;i<len\_word;i++)      {          if(strcmp(word[i], "kw") == 0){              fprintf(output\_file, "%s ", word[i+1]);              printf("%s ", word[++i]);          }      }      fprintf(output\_file,"\nFunctions: \t");      printf("\nFunctions: \t");      for(int i=0;i<number\_of\_value\_in\_symble\_table;i++)      {          if(strcmp(st[i].id\_type, "Func") == 0){              fprintf(output\_file, "%s ",st[i].name);              printf("%s ",st[i].name);          }      }      fprintf(output\_file, "\n");      printf("\n");      fclose(output\_file);  }  void remove\_unnecessary\_files()  {      remove("output\_assignment\_1.txt");      remove("output\_assignment\_2.txt");      remove("step2input.txt");      remove("with\_line.txt");      remove("without\_comment.txt");  }  int main(void)  {      take\_input();      call\_assignment1();      call\_assignment2();      make\_symble\_table();      check\_error();      remove\_unnecessary\_files();      return 0;  } |