2022/11/7

자료구조(01)

Programming Assignment Ⅱ

컴퓨터공학과

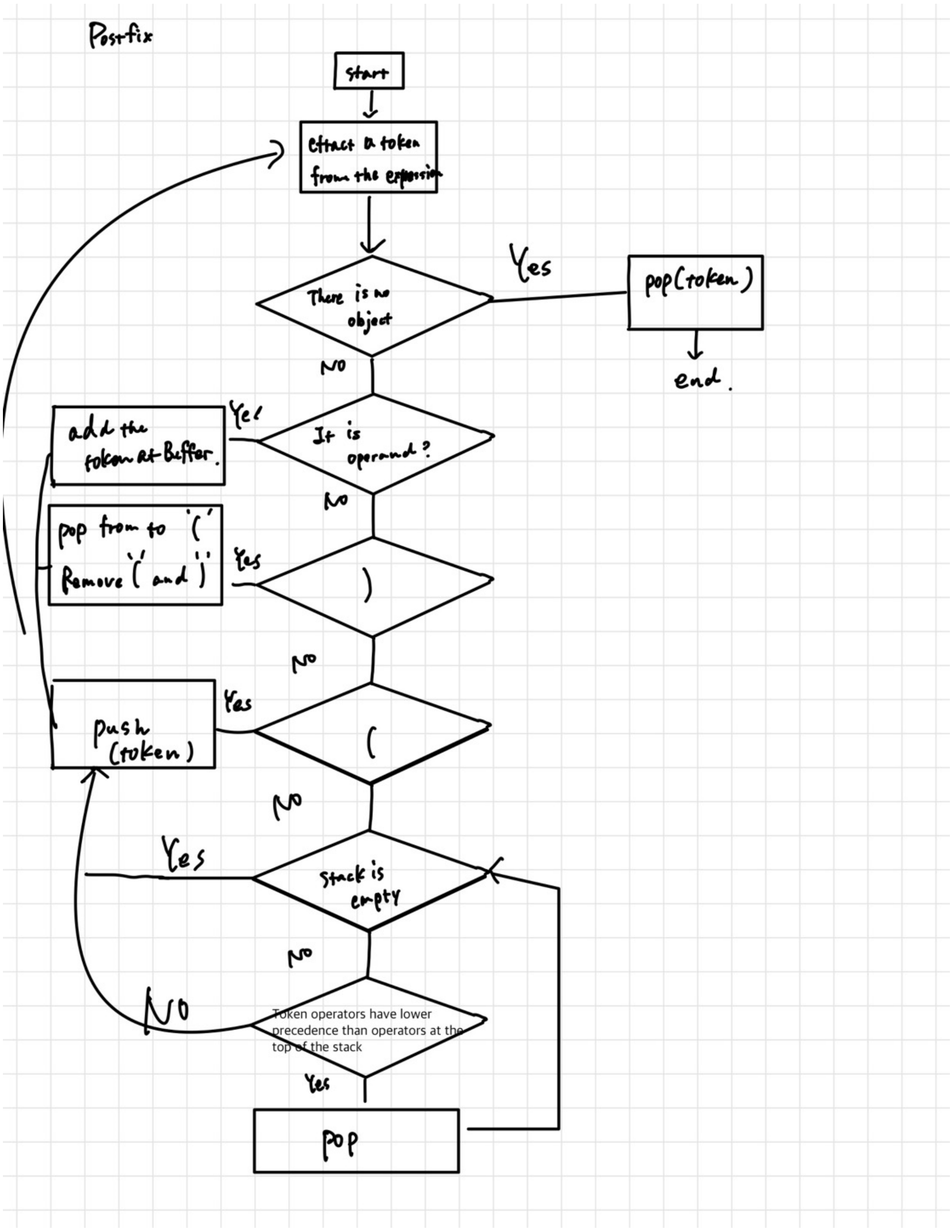
학번: 20202106

이름: OSHIMA ASUKA

자료구조 HW3
20202106 OSHIMA ASUKA

Question1：

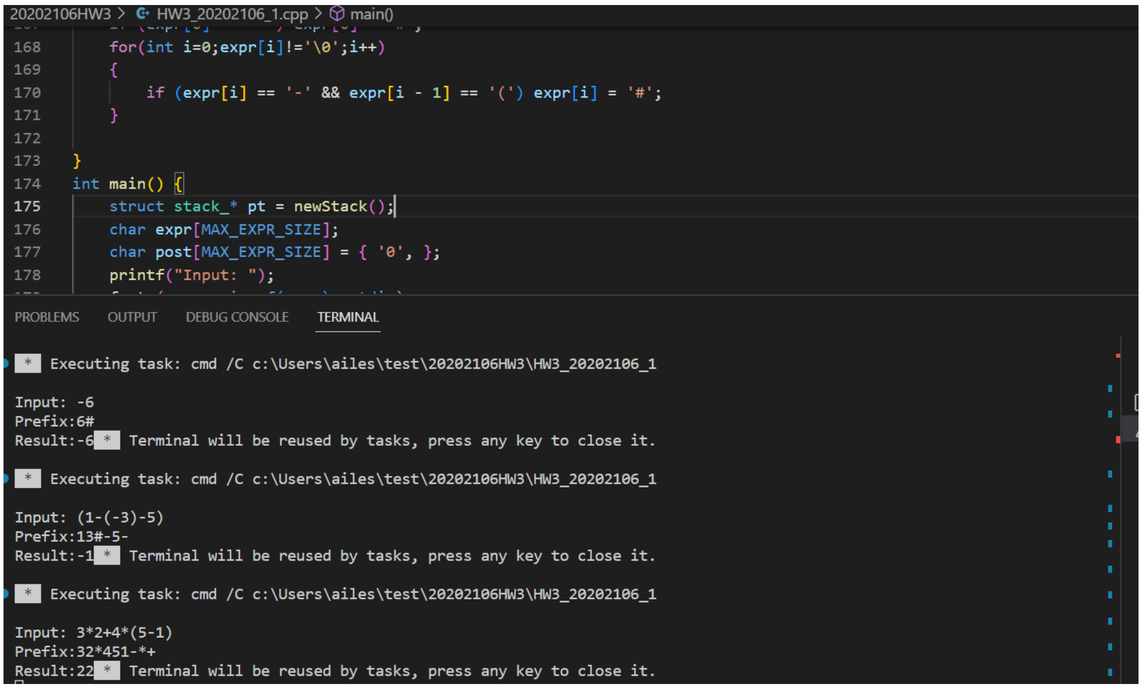
・Flowchart(q1)



・Pseudocode(q1)

```
【HW3 question1】
typedef enum {
   lparen, rparen, plus, minus, times, divide, mod, eos, unary, operand
}precedence;

struct stack_
::=
   int top;
   int* stack;
struct stack_* newStack()
::=  Create the newStack
   struct stack_* pt = (struct stack_*)malloc(sizeof(struct stack_));
   top := 0
   stack := (int*)malloc(sizeof(int) * MAX_STACK_SIZE);
   return pt

Boolean stackFull()
::=
   if(number of elements in stack == MAX_STACK_SIZE) return TRUE
   else return FALSE


void stackEmpty()
::=  if(stack==newStack(Size)) return TRUE
   else return FALSE

void push(precedence item, struct stack_* pt) ::=
   if (stack is Full) stackFull();
   else insert item into top of stack and return

int pop(struct stack_* pt) ::=
   if (top == -1) stackEmpty();
   else remove and return the item at the top pf the stack


void saveToken(precedence token, char* post,int in) ::=
     Store the token in post[in]
   switch (token) ~~~


precedence getToken(char* symbol, int* n, char* expr) ::=
   Get the next token, symbol is the character representation, which is returned, the
   token is represented by its enumerated value, which is returned in the function name.
```

```
void postfix(char* expr, struct stack_* pt, char* result)::=
   Get the postfix expression and save the result[]

   for token := getToken(&symbol, &n, expr) to  token is not NULL; token = getToken(&symbol, &n, expr
      if (token is not some operator)
         result[i]: = symbol

      else if (token == rparen)
          Unstack tokens until left parenthesis


         Discard the left parenthesis
      else

         Remove and print symbols whose isp is greater
         than or equal to the current token's icp

     push(token, pt)


  while ((token = (precedence)pop(pt)) != eos)
      saveToken(token, result, in)
      in += 1;

int eval( struct stack_* pt,char* post) ::=
   Evaluate a postfix expression, expr, '\0' is the end of the expression

    token = getToken(&symbol, &n, post);

        while (end of the expression )
                if (token == operand)
                        push((precedence)(symbol - '0'),pt)
                else
                        op2 = pop(pt)
                        op1 = pop(pt)
                        switch (token) :=
               Caluculate in each case and push the stack
                token = getToken(&symbol, &n, post)
        return caucualting result

int main() {
        struct stack_* pt = newStack();

        Input the string in expr[]
        postfix(expr, pt, post)
       change unary operator '-' to '#'
        output the postfix expression

        int result = eval(pt, post)
    Output the calculating result

}
```



20202106HW3 > HW3_20202106_1.cpp > main()

```
    for(int i=0;expr[i]!='\0';i++)
    {
        if (expr[i] == '-' && expr[i - 1] == '(') expr[i] = '#';
    }

}
int main() {
    struct stack_* pt = newStack();
    char expr[MAX_EXPR_SIZE];
    char post[MAX_EXPR_SIZE] = { '0', };
    printf("Input: ");
```

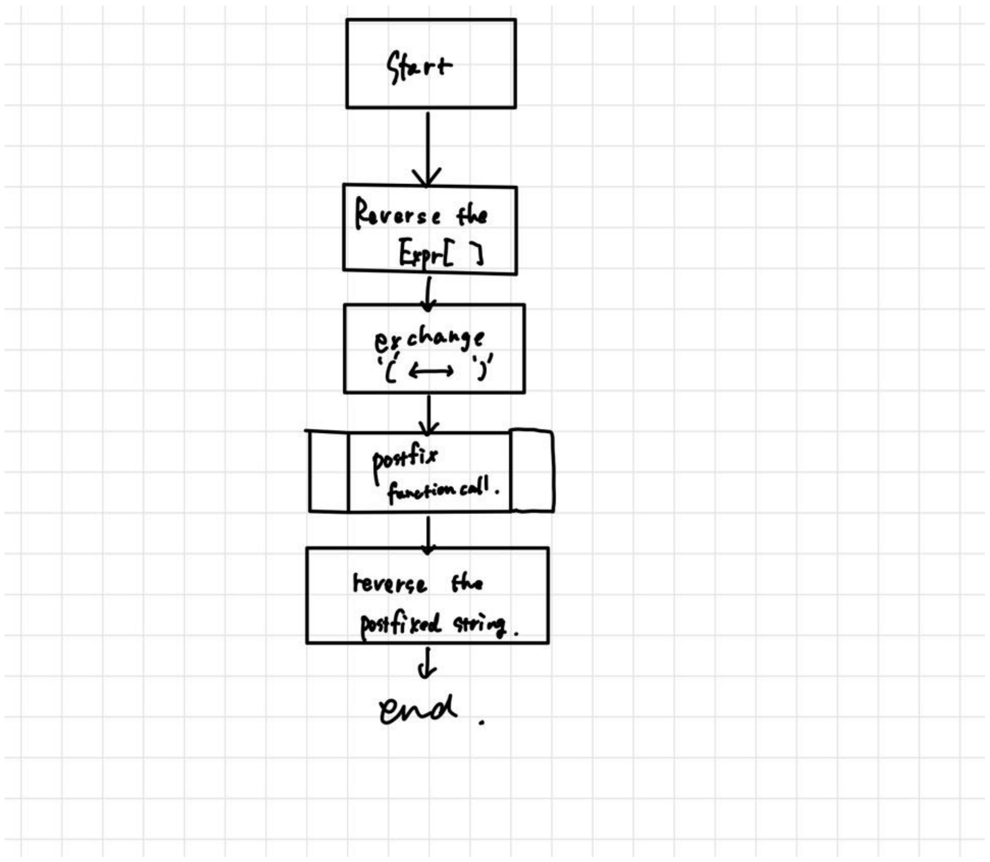
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
 *  Executing task: cmd /C c:\Users\ailes\test\20202106HW3\HW3_20202106_1

Input: -6
Prefix:6#
Result:-6 *  Terminal will be reused by tasks, press any key to close it.

 *  Executing task: cmd /C c:\Users\ailes\test\20202106HW3\HW3_20202106_1

Input: (1-(-3)-5)
Prefix:13#-5-
Result:-1 *  Terminal will be reused by tasks, press any key to close it.

 *  Executing task: cmd /C c:\Users\ailes\test\20202106HW3\HW3_20202106_1

Input: 3*2+4*(5-1)
Prefix:32*451-*+
Result:22 *  Terminal will be reused by tasks, press any key to close it.
```

자료구조 HW3
20202106 OSHIMA ASUKA

Question 2：

・Flowchart(q2)



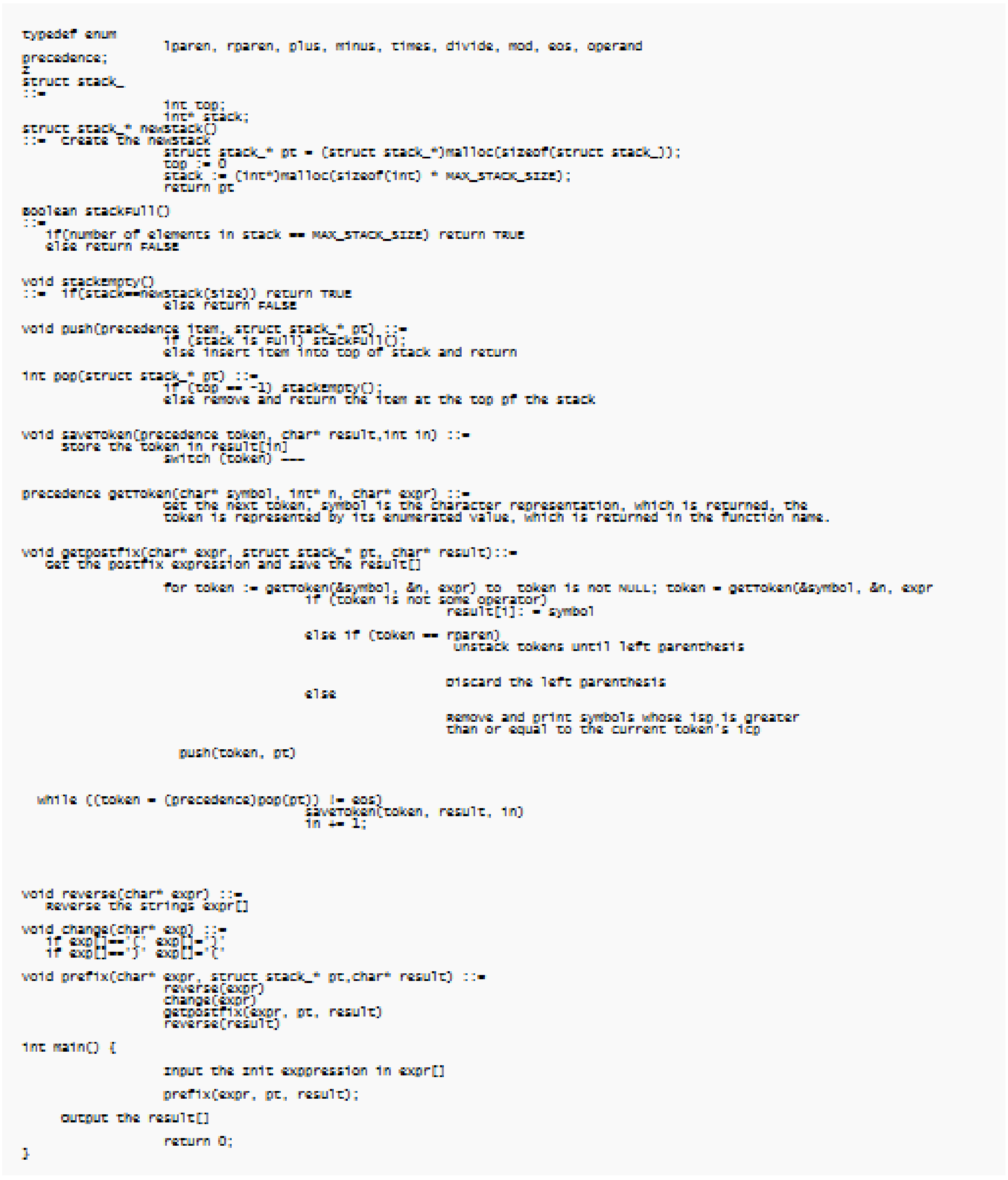
・Pseudocode(q2)

Figure is the preudocode of the question2. The point of how to get postfix exppresion is :

① Reverse the infix string and the string you must interchange left and right parentheses.

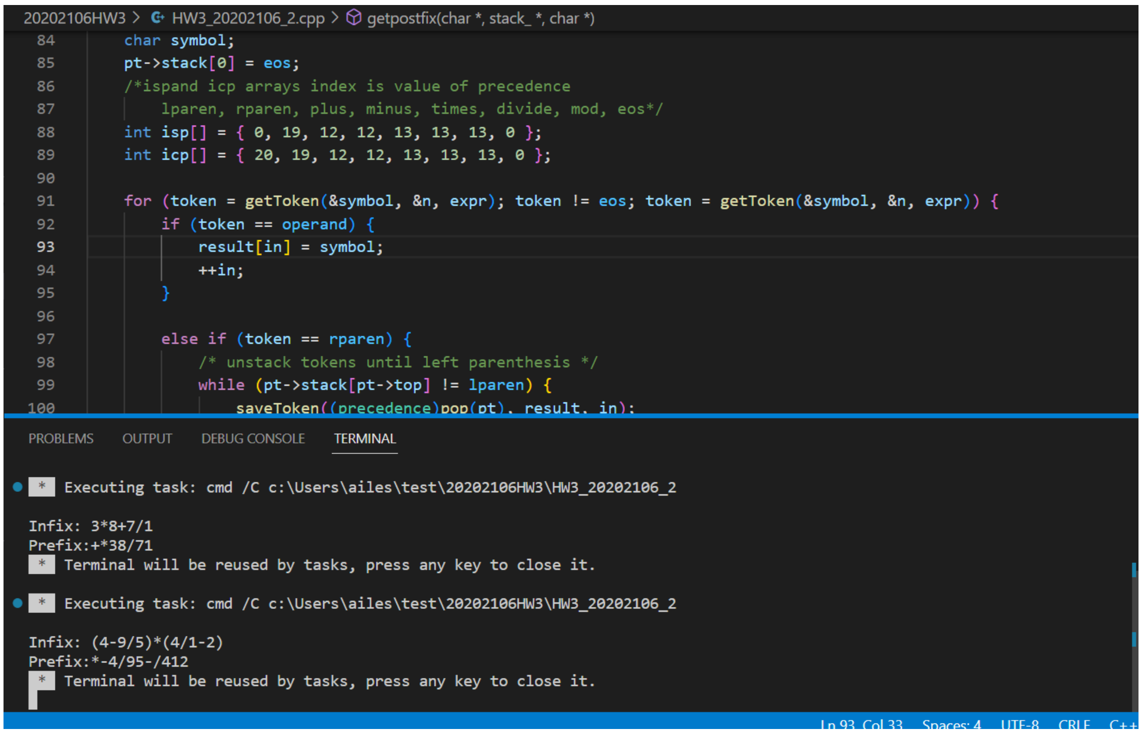
②Get the postfix expression of the infix expression ①

③Reverse the postfix expression to get the prefix expression.



```
typedef enum
                  lparen, rparen, plus, minus, times, divide, mod, eos, operand
precedence;
I
struct stack_
::=
                  int top;
                  int* stack;
struct stack_* newstack()
::=  create the newstack
                  struct stack_* pt = (struct stack_*)malloc(sizeof(struct stack_));
                  top := 0
                  stack := (int*)malloc(sizeof(int) * MAX_STACK_SIZE);
                  return pt

Boolean stackFull()
::=
   if(number of elements in stack == MAX_STACK_SIZE) return TRUE
   else return FALSE


void stackempty()
::=  if(stack==newstack(size)) return TRUE
                  else return FALSE

void push(precedence item, struct stack_* pt) ::=
                  if (stack is Full) stackFull();
                  else insert item into top of stack and return

int pop(struct stack_* pt) ::=
                  if (top == -1) stackempty();
                  else remove and return the item at the top pf the stack


void saveToken(precedence token, char* result,int in) ::=
     store the token in result[in]
                  switch (token) ---


precedence getToken(char* symbol, int* n, char* expr) ::=
                  Get the next token, symbol is the character representation, which is returned, the
                  token is represented by its enumerated value, which is returned in the function name.


void getpostfix(char* expr, struct stack_* pt, char* result)::=
   Get the postfix expression and save the result[]

                  for token := getToken(&symbol, &n, expr) to  token is not NULL; token = getToken(&symbol, &n, expr
                                  if (token is not some operator)
                                                  result[i]: = symbol

                                  else if (token == rparen)
                                                   unstack tokens until left parenthesis


                                                  Discard the left parenthesis
                                  else

                                                  Remove and print symbols whose isp is greater
                                                  than or equal to the current token's icp

                    push(token, pt)


  while ((token = (precedence)pop(pt)) != eos)
                                  saveToken(token, result, in)
                                  in += 1;




void reverse(char* expr) ::=
   Reverse the strings expr[]

void change(char* exp) ::=
   if exp[]=='(' exp[]=')'
   if exp[]==')' exp[]='('

void prefix(char* expr, struct stack_* pt,char* result) ::=
                  reverse(expr)
                  change(expr)
                  getpostfix(expr, pt, result)
                  reverse(result)

int main() {

                  Input the init exppression in expr[]

                  prefix(expr, pt, result);

     output the result[]

                  return 0;
}
```

Figure 1


20202106HW3 > HW3_20202106_2.cpp > getpostfix(char *, stack_ *, char *)
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Ln 93, Col 33
Spaces: 4
UTF-8
CRLF
C++

```
    char symbol;
    pt->stack[0] = eos;
    /*ispand icp arrays index is value of precedence
        lparen, rparen, plus, minus, times, divide, mod, eos*/
    int isp[] = { 0, 19, 12, 12, 13, 13, 13, 0 };
    int icp[] = { 20, 19, 12, 12, 13, 13, 13, 0 };

    for (token = getToken(&symbol, &n, expr); token != eos; token = getToken(&symbol, &n, expr)) {
        if (token == operand) {
            result[in] = symbol;
            ++in;
        }

        else if (token == rparen) {
            /* unstack tokens until left parenthesis */
            while (pt->stack[pt->top] != lparen) {
                saveToken((precedence)pop(pt), result, in);
```

```
 *  Executing task: cmd /C c:\Users\ailes\test\20202106HW3\HW3_20202106_2

Infix: 3*8+7/1
Prefix:+*38/71
 *  Terminal will be reused by tasks, press any key to close it.

 *  Executing task: cmd /C c:\Users\ailes\test\20202106HW3\HW3_20202106_2

Infix: (4-9/5)*(4/1-2)
Prefix:*-4/95-/412
 *  Terminal will be reused by tasks, press any key to close it.
```

Question3

DFS,Depth First Search, is an edge-based technique, and uses the Stack Data strucutre. The first step is that visited vertices are pushed into the stack. Second if there are no vertices then visited vertices are popped.[1] BFS, Breadth-First Search, is a vertex-based technique for finding the shortest path in the graph and uses a Queue data structure that follows first in first out. In BFS, when it tis visited and marked one vertex is selected at a time, then its adjacent are visited and stored in the queue. BFS is slower than DFS.

[1] **Difference between BFS and DFS,**
https://www.geeksforgeeks.org/difference-between-bfs-and-dfs/