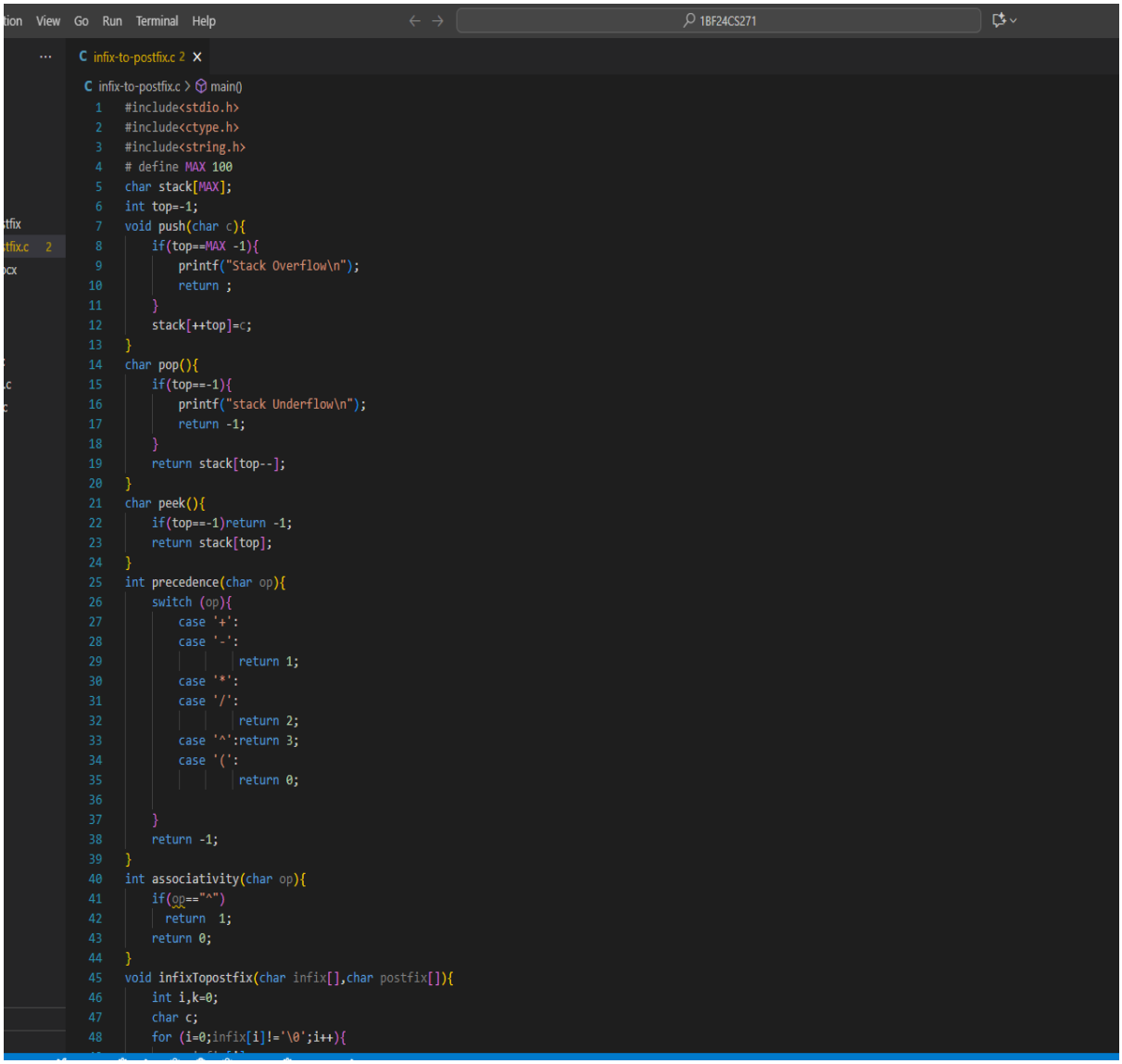
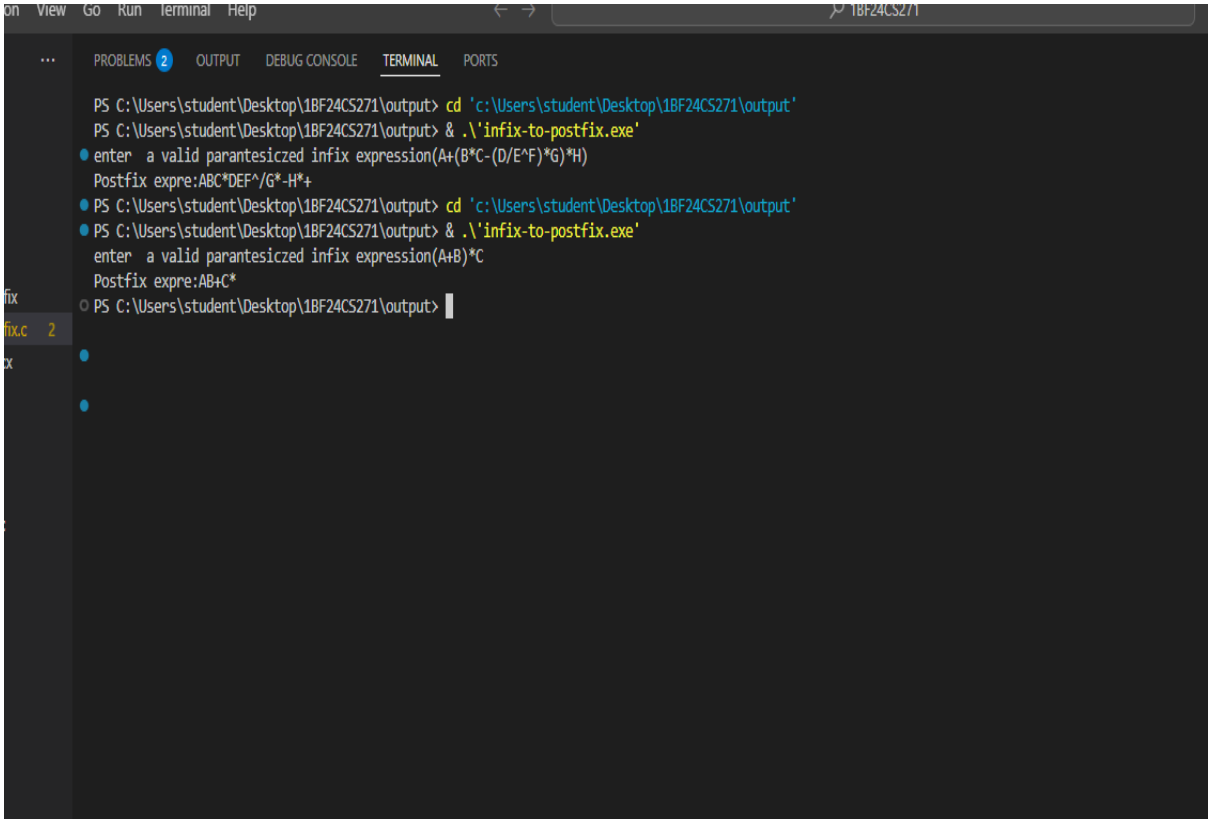
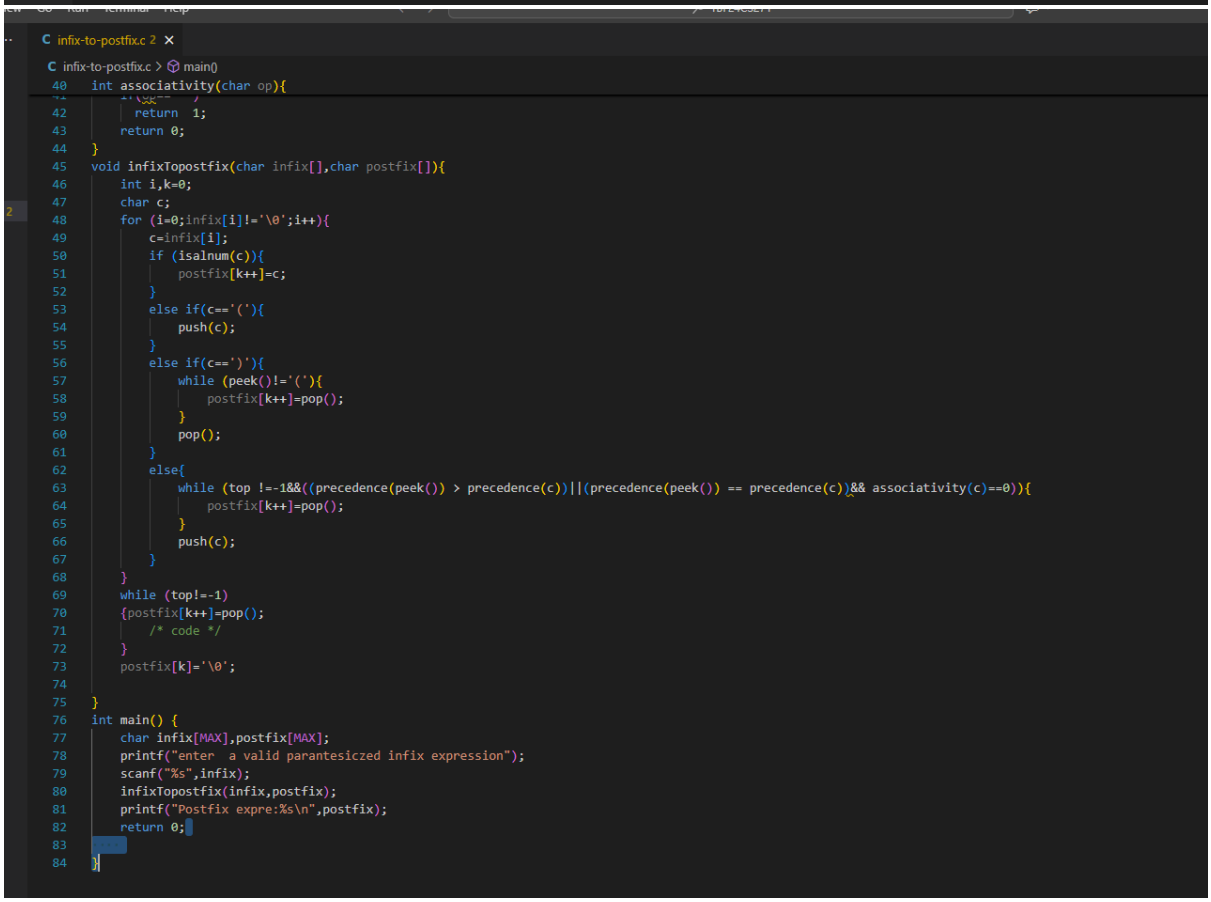

tion View Go Run Terminal Help
1BF24CS271
infix-to-postfix.c 2
infix-to-postfix.c > main()
#include<stdio.h>
#include<ctype.h>
#include<string.h>
define MAX 100
char stack[MAX];
int top=-1;
void push(char c){
if(top==MAX -1){
printf("Stack Overflow\n");
return ;
}
stack[++top]=c;
}
char pop(){
if(top==-1){
printf("stack Underflow\n");
return -1;
}
return stack[top--];
}
char peek(){
if(top==-1)return -1;
return stack[top];
}
int precedence(char op){
switch (op){
case '+':
case '-':
return 1;
case '*':
case '/':
return 2;
case '^':return 3;
case '(':
return 0;
}
return -1;
}
int associativity(char op){
if(op=="^")
return 1;
return 0;
}
void infixTopostfix(char infix[],char postfix[]){
int i,k=0;
char c;
for (i=0;infix[i]!='\0';i++){


View Go Run Terminal Help
1BF24CS271
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\student\Desktop\1BF24CS271\output> cd 'c:\Users\student\Desktop\1BF24CS271\output'
PS C:\Users\student\Desktop\1BF24CS271\output> & .\'infix-to-postfix.exe'
enter a valid parantesiczed infix expression(A+(B*C-(D/E^F)*G)*H)
Postfix expre:ABC*DEF^/G*-H*+
PS C:\Users\student\Desktop\1BF24CS271\output> cd 'c:\Users\student\Desktop\1BF24CS271\output'
PS C:\Users\student\Desktop\1BF24CS271\output> & .\'infix-to-postfix.exe'
enter a valid parantesiczed infix expression(A+B)*C
Postfix expre:AB+C*
PS C:\Users\student\Desktop\1BF24CS271\output>


infix-to-postfix.c 2
infix-to-postfix.c > main()
int associativity(char op){
return 1;
return 0;
}
void infixTopostfix(char infix[],char postfix[]){
int i,k=0;
char c;
for (i=0;infix[i]!='\0';i++){
c=infix[i];
if (isalnum(c)){
postfix[k++]=c;
}
else if(c=='('){
push(c);
}
else if(c==')'){
while (peek()!='('){
postfix[k++]=pop();
}
pop();
}
else{
while (top !=-1&&((precedence(peek()) > precedence(c))||(precedence(peek()) == precedence(c))&& associativity(c)==0)){
postfix[k++]=pop();
}
push(c);
}
}
while (top!=-1)
{postfix[k++]=pop();
/* code */
}
postfix[k]='\0';
}
int main() {
char infix[MAX],postfix[MAX];
printf("enter a valid parantesiczed infix expression");
scanf("%s",infix);
infixTopostfix(infix,postfix);
printf("Postfix expre:%s\n",postfix);
return 0;
}