Exp15 : Intermediate Code Generation

1. Start
2. Read the code in array – input
3. n = strlen(input)
4. i = n-1 , j = 0
5. while (i > 0)
    1. store input[i] , input[i-1] , input[i-2] in f
    2. if(input[i] == ')') then // to handle parenthesis when it is in last : a = b + (c * d)

        1. store input[i-1] , input[i-2] , input[i-3] in f
        2. print j,f
        3. i = i-2;
    3. else if(input[i-2] == ')') then // to handle parenthesis when it appears in other positions (i -2)
        1. store input[i-1] , input[i-2] , input[i-3] , input[i-4] , input[i-5] , input[i-6] in f
        2. print j , f[1] , f[2] , f[3]
        3. j++
        4. if (j-2 < 0) then // to handle code with two t statements : t1 = t0 + b

            print j , j-1 , f[5] , f[6]
        5. else // to handle code with complete(3) t : t2 = t1*t0

            print j , j-1 , f[5] , f[6] , j-2
        6. i = i-4;
    4. else if(i = 2 & f[1] = '=') then // if the statement is assignment type : a = b + c * d / f then to manage the assignment : a = t5

        1. print f[0],j-1
        2. break
    5. else if(i = (n-1)) then // first case (the first code that would be generated if there is no parenthesis : a + b * c then t0 = b * c

        1. print j,f
    6. else // to handle cases with two t : a + b * c / e and t0 = c / e
    then t1 = b * t0 , t2 = a + t1

        1. print j , f , j-1
    7. j++;
    8. i = i-2;
    9. input[i] = ‘t’
6. Stop

NOTE

This code only work if you use only two variables inside the parenthesis or do not using any parenthesis

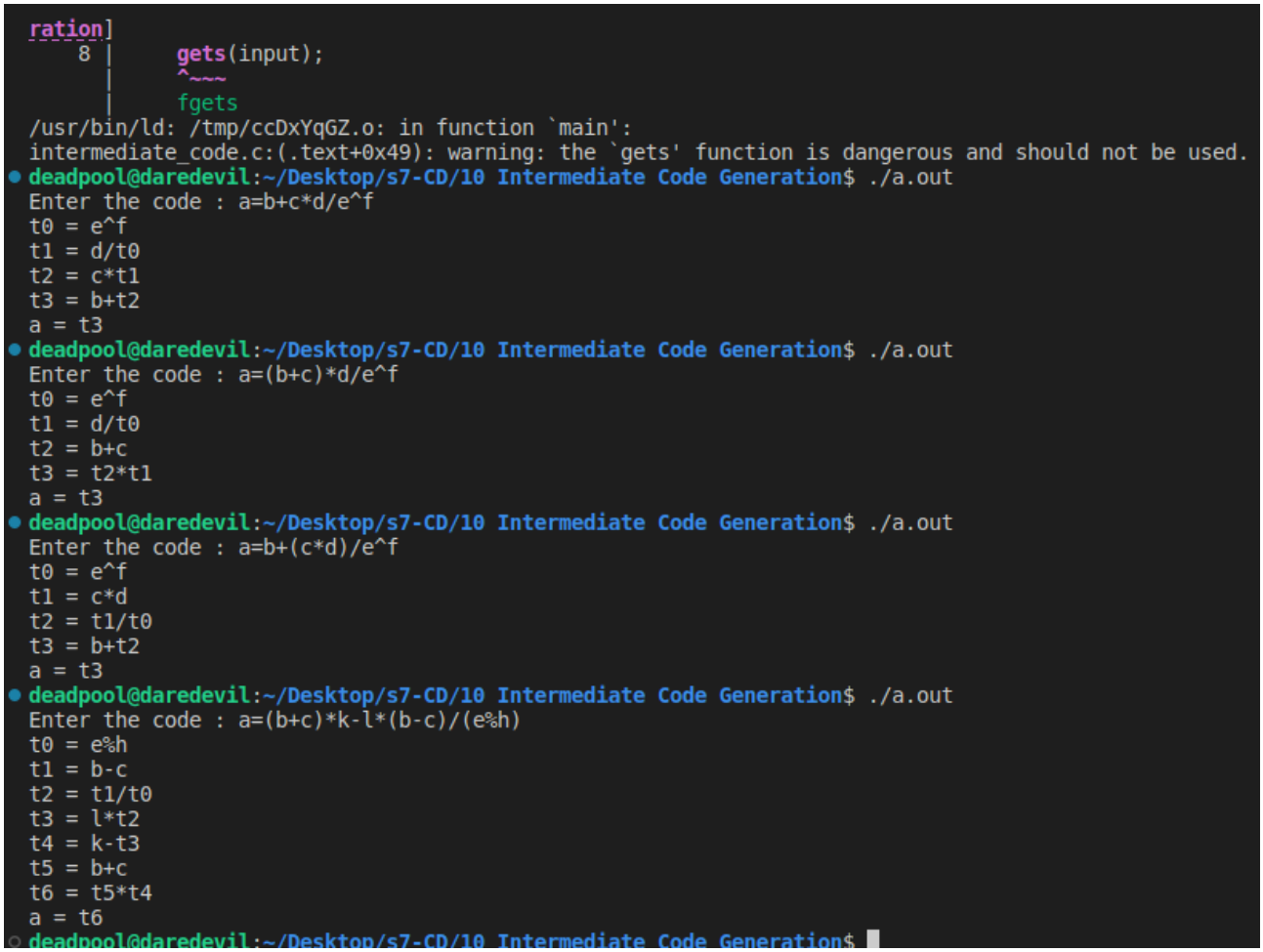
Intermediate Code Generation (C)

```
#include<stdio.h>
#include<string.h>

int main(){
   int n,i,j=0;
   char input[100],f[10];
   printf("Enter the code : ");
   gets(input);
   n=strlen(input);
   i=n-1;

   while(i>0){
      f[2] = input[i];
      f[1] = input[i-1];
      f[0] = input[i-2];
      if(input[i] == ')'){
         f[2] = input[i-1];
         f[1] = input[i-2];
         f[0] = input[i-3];
         printf("t%d = %s\n",j,f);
         i=i-2;
      }else if(input[i-2] == ')'){
         f[6] = input[i];
         f[5] = input[i-1];
         f[4] = input[i-2];
         f[3] = input[i-3];
         f[2] = input[i-4];
         f[1] = input[i-5];
         f[0] = input[i-6];
         printf("t%d = %c%c%c\n",j,f[1],f[2],f[3]);
         j++;
         if(j-2 < 0){
            printf("t%d = t%d%c%c\n",j,j-1,f[5],f[6]);
         }else{
            printf("t%d = t%d%c%c%d\n",j,j-1,f[5],f[6],j-2);
         }
         i=i-4;
      }else if(i == 2 && f[1] == '='){
         printf("%c = t%d\n",f[0],j-1);
         break;
      }else if(i == (n-1)){
         printf("t%d = %s\n",j,f);
      }else{
         printf("t%d = %c%c%c%d\n",j,f[0],f[1],f[2],j-1);
      }
      j++;
      i=i-2;
      input[i] = 't';
   }
   return 0;
}
```

output


ration]
8 | gets(input);
| ^~~~
| fgets
/usr/bin/ld: /tmp/ccDxYqGZ.o: in function `main':
intermediate_code.c:(.text+0x49): warning: the `gets' function is dangerous and should not be used.
deadpool@daredevil:~/Desktop/s7-CD/10 Intermediate Code Generation$ ./a.out
Enter the code : a=b+c*d/e^f
t0 = e^f
t1 = d/t0
t2 = c*t1
t3 = b+t2
a = t3
deadpool@daredevil:~/Desktop/s7-CD/10 Intermediate Code Generation$ ./a.out
Enter the code : a=(b+c)*d/e^f
t0 = e^f
t1 = d/t0
t2 = b+c
t3 = t2*t1
a = t3
deadpool@daredevil:~/Desktop/s7-CD/10 Intermediate Code Generation$ ./a.out
Enter the code : a=b+(c*d)/e^f
t0 = e^f
t1 = c*d
t2 = t1/t0
t3 = b+t2
a = t3
deadpool@daredevil:~/Desktop/s7-CD/10 Intermediate Code Generation$ ./a.out
Enter the code : a=(b+c)*k-l*(b-c)/(e%h)
t0 = e%h
t1 = b-c
t2 = t1/t0
t3 = l*t2
t4 = k-t3
t5 = b+c
t6 = t5*t4
a = t6
deadpool@daredevil:~/Desktop/s7-CD/10 Intermediate Code Generation$