

Tutorial 5: Searching and Hash Tables

- Q1** Given an array of n elements. Find two elements in the array such that their sum is equal to K . The two elements can be the same element. Once a pair of elements is found, the program can be terminated. The function prototype is given below:

```
void dualSearch(int A[], int size, int K, int dualIndex[])
```

- Q2** Given a sorted array of n elements. Find two elements in the array such that their sum is equal to K . The two elements can be the same element. Once a pair of elements are found, the program can be terminated. The results may be different from the results of Question 1. The function prototype is given below:

```
void dualSearch(int A[], int size, int K, int dualIndex[])
```

- Q3** Compare the performance between Q1 and Q2. Try to use a large input file to evaluate their running time. If you are using Code::Blocks, you may use *Configure Tools* under *tools* to create a User-defined tools shown in Figure 5.1. You can also try to run your programs in the command line or terminal directly using I/O redirection “<” for standard input and “>” for standard output. A 500k data and 1 million data file are attached. The first line is a search key and the second line is a data size. The rest are the data.

- Q4** Implement a closed addressing hash table to perform insertion and key searching. The insertion may not have to insert at the end of the link-list. The function prototype is given below:

```
ListNode* HashSearch(HashTable *, int);  
int HashInsert(HashTable *, int);
```

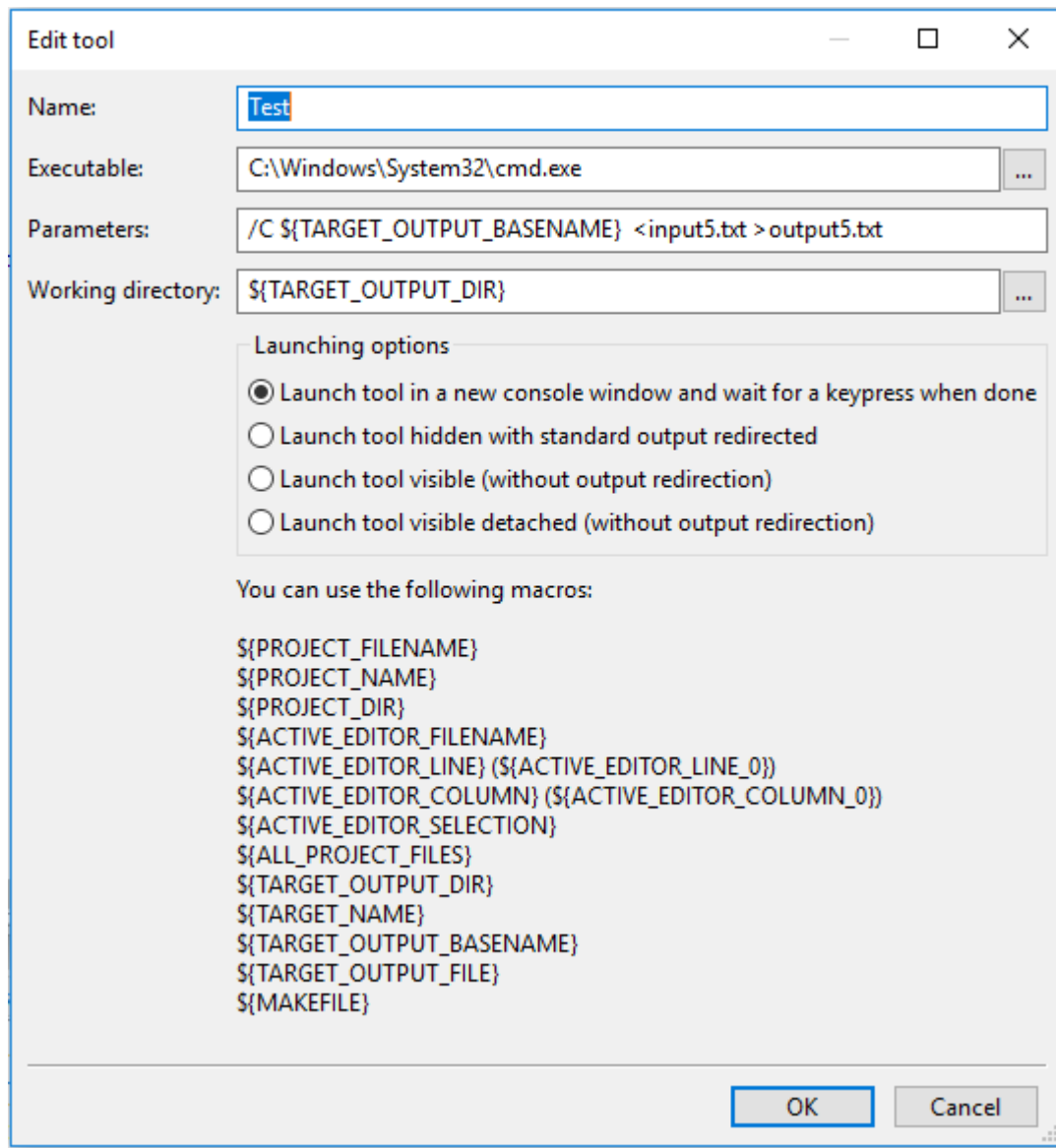


Figure 5.1: Name is not important. It can be anything. `input5.txt` is an input filename and `output5.txt` is an output filename.