1

LAB13 – Command-Line Arguments Exercise 1

- Write a program to calculate the **average** of integer numbers that are given as **command-line arguments** by the user.
- Program should determine how many integers entered from the command-line.
- In the example below, the user inputs are shown in red color.

1. Press Win+R keys
2. Write CMD
3. Press ENTER key
4. Write "cd Documents"

Command-line Window

```
C:\Documents> exercise1.exe  15  10  27  34

15
10
27
34

Average = 21.500000
```

2

Input/Output Redirection in Command-line

3

Input/Output Redirection

- Redirecting Input/Output on UNIX and Windows Systems
  - Standard input device is keyboard
  - Standard output device is screen
  - User can redirect the input and/or output to/from files, when a program is executed from command-line window

4

14.2 Redirecting Input

- **Redirect input symbol (<)**
  - Determines what input device of a program is
  - Operating system feature, not a C feature
  - Example:

```
$ myprog < myinput.txt
```

  - Rather than inputting values by hand from keyboard, program gets them from `myinput.txt` file
  - $ or % represents the command line prompt symbol in Unix/Linux.
  - `C:\>` is the prompt symbol in Windows.

5

14.2 Redirecting Output

- **Redirect output (>)**
  - Determines where output of a program goes
  - Example:

```
$ myprog > myout.txt
```

  - Instead of screen, output goes into `myout.txt` (erases previous file contents if any)

6

14.2 Redirecting Both Input and Output

- We can use both < , and > redirections at the same time.
- Example:

```
$ myprog  <myinput.txt  >myout.txt
```

7

14.2 Redirecting and Appending Output

- **Append output (>>)**
  - Add output to end of file (preserve previous contents)
  - Example:

```
$ myprog >> myout.txt
```

  - Output is added onto the end of `myout.txt`

8

14.2 Piping Input and Output

- **Pipe command (|)**
  - Output of one program becomes input of another
  - Example:

```
$ prog1 | prog2
```

  - Output of `prog1` goes to `prog2` as input

9

Example: Two separate programs

prog1.c

```
#include <stdio.h>
int main() {
  int i;
  for (i=1; i <=5; i++)
       printf("%d\n",i*10);
}
```

prog2.c

```
#include <stdio.h>
int main() {
  int i,num,Tot=0;

  for (i=1; i <=5; i++)  {
       printf("\n Enter a number :");
       scanf("%d",&num);
       Tot += num;
  }

  printf("Average is : %.2f\n", Tot/5.0);
}
```

10

Exercise 2

- In command-line window, test the following.

```
C:\Documents> prog1.exe >> output1.txt
```

11

Exercise 3

- In command-line window, test the following.

```
C:\Documents> prog2.exe << output1.txt
```

12

Exercise 4

- In command-line window, test the following.

```
C:\Documents> prog1.exe  |  prog2.exe
```

Screen output

```
Enter a number :
Enter a number :
Enter a number :
Enter a number :
Enter a number :
Average is : 30.00
```