|  |  |  |
| --- | --- | --- |
| Test Case | Reason | Expected Outcome |
| abc + abc | Verifying the addition of two equal size strings | The program prints the input as well as the sum which is bdf  ./a.out abc + abc  abc + abc => bdf |
| abc + az | Verifying that the addition of two unequal strings will result in a string of the larger size (when the first string is the larger one).  Also verifying that the char in the first string is printed as a capitol if the sum is over 26 (z). | The program prints the input as well as the sum which is bBc  ./a.out abc + az  abc + az => bBc |
| abc + azzzz | Verifying that the addition of two unequal strings will result in a string with the larger size (when the second string is the larger one). | The program prints the input as well as the sum which is bBCzz  ./a.out abc + azzzz  abc + azzzz => bBCzz |
| fff - abc | Verifying the subtraction of two equal size strings | The program prints the input as well as the difference which is edc  ./a.out fff – abc  fff - abc => edc |
| fff - az | Verifying that the subtraction of two unequal size strings will result in a string of the larger size (when the first string is the larger one). | The program prints the input as well as the difference which is eFf  ./a.out fff – az  fff - az => eFf |
| fff - azssss | Verifying that the subtraction of two unequal size strings will result in a string of the larger size (when the second string is the larger one). | The program prints the input as well as the difference which is eFFsss  ./a.out fff – azssss  fff - azssss => eFFsss |
| fff / abg | Verifying the division of two equal size strings | The program prints the input as well as the quotient which is fcF  ./a.out fff / abg  fff / abg => fcF |
| fff / az | Verifying that the division of two unequal size strings will result in a string of the larger size (when the first string is the larger one). | The program prints the input as well as the quotient which is fFf  ./a.out fff / az  fff / az => fFf |
| fff / azfghy | Verifying that the division of two unequal size strings will result in a string of the larger size (when the second string is the larger one). | The program prints the input as well as the quotient which is fFaghy  ./a.out fff / azfghy  fff / azfghy => fFaghy |
| abc x abd | Verifying the multiplication of two equal size strings. | The program prints the input as well as the product which is adl  ./a.out abc x abd  abc x abd => adl |
| abcc x zbz | Verifying that the multiplication of two unequal size strings will result in a string of the larger size (when the first string is the larger one). | The program prints the input as well as the product which is zdCc  ./a.out abcc x zbz  abcc x zbz => zdCc |
| adv x zzbbb | Verifying that the multiplication of two unequal size strings will result in a string of the larger size (when the second string is the larger one). | The program prints the input as well as the product which is zDVbb  ./a.out adv x zzbbb  adv x zzbbb => zDVbb |
| a+ b | Verifying that the user cannot enter an incorrect number of arguments.  In this case, there is no space after the first argument so only a total of two arguments are seen by the program. | The program lets the user know that they did not enter the correct number of arguments.  ./a.out a+ b  bad input |
| Fxx + gf | Verifying that the user cannot enter a capitol letter. | The program lets the user know that they input bad values  ./a.out Fxx + gf  bad input |

EXTRA CREDIT

|  |  |  |
| --- | --- | --- |
| abc ^ abc | Verifying the power function for two equal length strings. | The program prints the input as well as the result of the power  ./a.out abc ^ abc  abc ^ abc => adC |
| abcccc ^ abb | Verifying the power function where the first string is longer than the second. | The program prints the input as well as the result of the power  ./a.out abcccc ^ abcd  abcccc ^ abcd => adCCcc |
| abc ^ abccc | Verifying the power function where the second string is shorter than the first. | The program prints the input as well as the result of the power  ./a.out abc ^ abccc  abc ^ abccc => adCcc |