1. You are required to fix all logical errors in the given code. You can click on compile & Run anytime to check the compilation / execution status of the program. You can use print(f) to debug your code. The submitted code should be logically / syntactically correct and pass all testcases.

**Code Approach :** For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach.

The function/method **patternPrint** accepts an argument num, an integer

The function/method **patternPrint** prints num lines in the following pattern.

For Example num -= 4, the pattern should be

1

11

111

1111

The function/method **patternPrint** compiles successfully but fails to print the desired result for some test cases due to incorrect implementation of the function/method. Your task is to fix the code so that it passes all the test cases.

1. // You can print the values to stdout for debugging
2. void patternPrint(int num)
3. {
4. int print=1.i,j)
5. for (i=0;1<num;1++)
6. {
7. for (j=0; j<=i; j++)
8. {
9. Printf(“%d”, print);
10. **}**
11. Printf(“\n”);
12. **}**
13. **}**

**Console Output = Compilation Successful**

1. You are required to fix all logical errors in the given code. You can click on compile & Run anytime to check the compilation / execution status of the program. You can use print(f) to debug your code. The submitted code should be logically / syntactically correct and pass all testcases. Do not write the main() function as it is not required.

**Code Approach :** For this question, you will need to complete the code as in given implementation. We **do not** expect you to modify the approach.

The function/method **patternPrint** accepts an argument num, an integer

The function/method **patternPrint** prints num lines in the following pattern.

For Example num -= 4, the pattern should be

1

11

111

1111

The function/method **patternPrint** compiles successfully but fails to print the desired result for some test cases due to incorrect implementation of the function/method. Your task is to fix the code so that it passes all the test cases.

1. // You can print the values to stdout for debugging
2. void patternPrint(int num)
3. {
4. int print=1.i,j)
5. for (i=0;1<num;1++)
6. {
7. for (j=0; j<=i; j++)
8. {
9. Printf(“%d”, print);
10. **}**
11. Printf(“\n”);
12. **}**
13. **}**

**Console Output = Compilation Successful**

1. You are required to fix all syntactical errors in the given code. You can click on compile & Run anytime to check the compilation / execution status of the program. You can use print(f) to debug your code. The submitted code should be logically / syntactically correct and pass all testcases. Do not write the main() function as it is not required.

**Code Approach :** For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library methods.

The function/method **matrixSum** returns an integer representing the sum of elements of the input matrix. The function/method **matrixSum** accepts three arguments – rows, an integer representing the number of rows of the input matrix, columns, an integer representing the number of columns of the input matrix and matrix, a two-dimensional array representing the input …...

The function/method **matrixSum** compiles unsuccessfully due to debug the program so that it passes all test cases.

1. // You can print the values to stdout for debugging
2. int matrixsum(int rows, int columns, int \*\*matrix)
3. {
4. int i, j sum = 0
5. for (i=0;1<rows;1++)
6. {
7. for (j=0; j<columns; j++)
8. sum += matrix[i][j];
9. }
10. Return sum;
11. }

**Console Output = Compilation Successful**

1. You are required to complete the given code. You can click on compile & Run anytime to check the compilation / execution status of the program. You can use print(f) to debug your code. The submitted code should be logically / syntactically correct and pass all testcases. Do not write the main() function as it is not required.

**Code Approach :** For this question, you will need to complete the code as in given implementation. We **do not** expect you to modify the approach or incorporate any additional library methods.

The function/method **allExponent** returns a real number representing the result of exponentiation of base raised to power exponent for all input values. The function/method **allExponent** accepts two arguments baseValue, an integer representing the base and exponentValue, an integer representing the exponent.

The incomplete code in the function/method **allExponent** works only for positive values of the exponent. You must complete the code and make it work for negative values of exponent as well.

Another function/method **positiveExponent** uses an efficient way for exponentiation but accepts any positive exponent values. You are supposed to use this function/method to complete the code.

In **allExponent** function/method **Helper Description**

The following function is used to represent a positiveExponent and is already implemented in the default mode (Do not write the definition again in your code)

int positiveExponent(int baseValue, int,

{

It calculates the Exponent for the

This can be called as

int ces – (float)positiveExponent

1. // You can print the values to stdout for debugging
2. float allExponent(int baseValue, int exponentValue)
3. {
4. float res = 1;
5. If(exponentValue >=0)
6. {
7. res = (float)positiveExponent(baseValue, exponentValue)
8. }
9. else
10. {
11. // write your code here for neragive value of exponentVelue
12. res =1/(float)positiveExponent(baseValue, exponentValue);
13. }
14. return res;
15. }

**Console Output = Compilation Successful**

1. You are required to fix all logical errors in the given code. You can click on compile & Run anytime to check the compilation / execution status of the program. You can use print(f) to debug your code. The submitted code should be logically / syntactically correct and pass all testcases.

**Code Approach :** For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library methods.

The function/method **selectionSortArray** performs an in-place selection sort on the given input list which will be sorted in ascending order. The function/method **selectionSortArray** accepts two arguments len, an integer representing the length of the input list and arr, a list of integers representing the input list respectively.

The function/method **selectionSortArray** complies successfully but fails to get the desired result for some test cases due to logical errors. Your task is to fix the code so that it passes all the test cases.

**Note :** In this particular implementation of selection sort, the smallest element in the list is swapped with the element at first index, the next smallest element is swapped with the element at the next index and so on.

1. // You can print the values to stdout for debugging
2. void selectionSortArray(int len, int\* arr)
3. {
4. Int x = 0, y = 0;
5. for (x=0; x<len; x++){
6. int index\_of\_min = x;
7. for (y=x; y<len; y++){
8. If(arr[index\_of\_min]>arr[y[){
9. Index\_of\_min = y;
10. }
11. }
12. int team = arr[x];
13. arr[x] = arr[index\_of\_min];
14. arr[index\_of\_mine] = temp;
15. }
16. }
18. You are required to fix all logical errors in the given code. You can click on compile & Run anytime to check the compilation / execution status of the program. You can use print(f) to debug your code. The submitted code should be logically / syntactically correct and pass all testcases.

**Code Approach :** For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library methods.

The function/method **descendingSortArray** performs an in-place selection sort on the given input list which will be sorted in descending order. The function/method **descendingSortArray** accepts two arguments len, an integer representing the length of the input list and arr, a list of integers representing the input list respectively.

The function/method **descendingSortArray** complies successfully but fails to get the desired result for some test cases due to logical errors. Your task is to fix the code so that it passes all the test cases.

1. // You can print the values to stdout for debugging
2. void descendingSorArray(int len, int\* arr)
3. {
4. int small, pos, i, j, temp;
5. for (i=0; i<=len-1; i++){
6. for (j-i; j<len;j++){
7. temp = 0;
8. if(arr[i]<arr[j]){
9. temp=arr[i];
10. arr[i]=arr[j];
11. arr[j]=temp;
12. }
13. }
14. }
15. }
16. You are required to fix all logical errors in the given code. You can click on compile & Run anytime to check the compilation / execution status of the program. You can use print(f) to debug your code. The submitted code should be logically / syntactically correct and pass all testcases. Do not write the main() function as it is not required.

**Code Approach :** For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library methods.

Lisa always forgets her birthday which is on the 5th July. So develop a function/method which will be helpful to remember her birthday.

The function/method **checkBirthDate** returns an integer ‘1’ if it is her birthday else returns 0. The function/method **checkBirthDate** accepts two arguments month, a string representing the month of her birthday and day, an integer representing the date of her birthday.

The function/method **checkBirthDate** complies successfully but fails to get the desired result for some test cases due to logical errors. Your task is to fix the code so that it passes all the test cases.

1. // You can print the values to stdout for debugging
2. int checkBirthDay(char\* month, int day)
3. {
4. if (Istrcmp(month, “July”}==0 && (day==5))
5. return1;
6. else
7. Return 0;
8. }
9. You are required to complete the given code. You can click on compile & Run anytime to check the compilation / execution status of the program. You can use print(f) to debug your code. The submitted code should be logically / syntactically correct and pass all testcases. Do not write the main() function as it is not required.

**Code Approach :** For this question, you will need to complete the code as in given implementation. We **do not** expect you to modify the approach or incorporate any additional library methods.

The function/method **findMaxElement** returns an integer representing the largest element in the given two input lists.

The function/method **findMaxElement** accepts four arguments len1, an integer representing the length of the first list, arr1, a list of integers representing the first input list, len2, an integer representing the length of the second input list and arr2, a list of integers representing the second input list respectively.

Another function/method **sortArray** accepts two argument, len an integer representing the length of the list and arr, a list of integers representing and return a list sorted ascending order.

Your task is to use the function/method **sortArray** to complet the code in **findMaxElement** so that it passes all the test cases.

1. // You can print the values to stdout for debugging
2. int\* sortArray(int len, int\* arr)
3. {
4. int i=0,j=0,temp=0;
5. for(i=0;i<len;i++)
6. {
7. for(j=i+1;j<len;j++)
8. {
9. if(arr[i]>arr[j]
10. {
11. temp = arr[i];
12. arr[i] = arr[j];
13. arr[j] = temp;
14. }
15. }
16. }
17. return arr;
18. }
20. int findMaxElement(int len1, int\* arr1, int len2, int\* arr2)
21. {
22. int \*arr3, \*arr4;
23. arr3 = sortArray(len1, arr1);
24. Arr4= sortArray(len1, arr2);
25. if(arr3[len1-1] > arr4[len2-1])
26. {
27. return arr3[len1-1];
28. }
29. else{
30. returnarr4[len2-1];
31. }
32. }
34. You are required to fix all logical errors in the given code. You can click on compile & Run anytime to check the compilation / execution status of the program. You can use print(f) to debug your code. The submitted code should be logically / syntactically correct and pass all testcases. Do not write the main() function as it is not required.

**Code Approach :** For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library methods.

The function/method **sumElement** returns an integer representing the sum of the element in the input array that are greater than twice the input numbers K and present at the even index.

The function/method **sumElement** accepts three arguments – size, an integer representing the size of the input array, numK, an integer representing the input number K and inputArray, a list of integers representing the input array. The function/method **sumElement** complies successfully but fails to get the desired result for some test cases due to logical errors.

**Note :** Index of the input array starts from 0.

1. int sumElement(int size, int numK, int \*inputArray)
2. {
3. int i, sum=0;
4. for (i=0; i<size; i++)
5. {
6. if(inputArray[i]>2\*numK && i % 2 == 0
7. {
8. sum += inputArray[i];
9. }
10. }
11. return sum;
12. }
14. You are required to fix all logical errors in the given code. You can click on compile & Run anytime to check the compilation / execution status of the program. You can use print(f) to debug your code. The submitted code should be logically / syntactically correct and pass all testcases.

**Code Approach :** For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library methods.

The function/method **arrayReverse** modify the input list by reversing its element.

The function/method **arrayReverse** accepts two arguments – len, an integer representing the length of the list and arr, list of integers representing the input list, respectively.

For Example : if the input list arr is (20 30 10 40 50) the function/method is supposed to print (50 40 10 30 20)

The function/method **arrayReverse** compiles successfully but fails to get the desired result for some test cases due to logical errors. Your task is to fix the code so that it passes all the test cases.

1. // You can print the values to stdout for debugging
2. void arrayReverse(int len, int\* arr)
3. {
4. Int i, temp, originalLen=len;
5. for (i=0; i<originalLen/2;i++){
6. temp = arr[len – 1]
7. arr[len – 1] = arr[i];
8. len -=1;
9. }
10. }

**Console Output = Compilation Successful**

1. You are required to fix all logical errors in the given code. You can click on compile & Run anytime to check the compilation / execution status of the program. You can use print(f) to debug your code. The submitted code should be logically / syntactically correct and pass all testcases.

**Code Approach :** For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library methods.

The function/method **countDigits** return an integer representing the remainder when the given number is divided by the number of digits in it.

The function/method **countDigits** accepts an argument – num, an integer representing the given number.

The function/method **countDigits** compiles successfully but fails to print the desired result for some test cases due to logical errors. Your task is to fix the code so that it passes all the test cases.

1. // You can print the values to stdout for debugging
2. int count Digits(int num){
3. int count = 0’
4. int num1 = num;
5. While( num != 0 ){
6. num = num / 10;
7. Count ++;
8. }
9. return (num1 % count );
10. }

**Console Output = Compilation Successful**

1. You are required to fix all logical errors in the given code. You can click on compile & Run anytime to check the compilation / execution status of the program. You can use print(f) to debug your code. The submitted code should be logically / syntactically correct and pass all testcases. Do not write the main() function as it is not required.

**Code Approach :** For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library methods.

The function/method **removeElement** prints space separated integers that remains after removing the integer at the given index from the input lists.

The function/method **removeElement** accepts three arguments, size an integer representing the size of the input list, indexValue, an integer representing given index and inputList, a list of integers representing the input lists.

The function/method **removeElement** compiles successfully but fails to print desired result for some test cases due to incorrect implementation of the function/method **removeElement**. Your task is to fix the code so that it passes all the test cases.

**Note :** zero ………… indexing is followed to access list elements.

1. // You can print the values to stdout for debugging
2. void removeElement(int size, int indexValue, int \*inputLists)
3. {
4. int i, j;
5. If(indexValue<size)
6. {
7. for(i=indexValue; i<size-1; i++)
8. {
9. inputList[i]=inputList[i+1];
10. }
11. for(i=0;i<size-1; i++)
12. Printf(“%d”, inputList[i]);
13. }
14. Else
15. {
16. for(i=0; i<size; i ++)
17. Printf(“%d”, inputList[i]);
18. }
19. }

**Console Output = Compilation Successful**

1. You are required to fix all syntactical errors in the given code. You can click on compile & Run anytime to check the compilation / execution status of the program. You can use print to debug your code. The submitted code should be logically / syntactically correct and pass all testcases. Do not write the main() function as it is not required.

**Code Approach :** For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library methods.

The function/method **countElement** return an integer representing the number of elements in the input list whichare greater than twice the input number K.

The function/method **countElement** accepts three arguments, size, an integer representing the size of the input list, mi,ner L and integer representing given input number K and inputList, a list of integers representing the input list respectively.

The function/method **countElement** compiles successfully due to symantic effort. Your task is to fix the code so that it passes all the test cases.

1. // You can print the values to stdout for debugging
2. Int countElement(int size, int numK, int \*inputList)
3. {
4. int i, count=0;
5. For(i=0; 1<size; i++)
6. {
7. If(inputList[1]>2\*numK)
9. Count+=1;
10. }
11. return count;
12. }

**Console Output = Compilation Successful**