ASSIGNMENT

User Defined Functions

MAY 3, 2023

MD ASADUZZAMAN ATIK (2023-1-60-130)
2023-1-60-130@std.ewubd.edu

Table of Contents

File: 09. Function_related_problems.docx

Function Related Problems (Total 20 questions)

1. Function to print a custom message.

   Code:

```
#include <stdio.h>

void custom_message(void);

int main(int argc, char *argv[]) {

    custom_message();

    return 0;

}

void custom_message(void) {

    printf("This is a function\n");

    return;

}
```

   Console:

```
This is a function

Process returned 0 (0x0)   execution time : 0.048 s
Press any key to continue.
```

2. Function to print an input character value.

   Code:

```
#include <stdio.h>

void print_char(char c);

int main(int argc, char *argv[]) {
    char input;

    scanf("%c", &input);

    print_char(input);

    return 0;

}

void print_char(char c) {

    printf("Value received from main: %c\n", c);

    return;

}
```

Console:

- For input: 3

```
3
Value received from main: 3

Process returned 0 (0x0)   execution time : 1.721 s
Press any key to continue.
```

- For input: A

```
A
Value received from main: A

Process returned 0 (0x0)   execution time : 0.690 s
Press any key to continue.
```

3. Function to calculate the sum of n numbers coming from the console.

Code:

```
#include <stdio.h>

int sum_n_numbers(int numberOfElements);
```

```c
int main(int argc, char *argv[]) {

    int n;

    scanf("%d", &n);

    int totalSum = sum_n_numbers(n);

    printf("Sum In Main: %d\n", totalSum);

    return 0;

}

int sum_n_numbers(int numberOfElements) {

    int sum = 0;

    for (size_t i = 0; i < numberOfElements; i++) {
        int number;
        scanf("%d", &number);
        sum += number;
    }

    printf("Sum In Function: %d\n", sum);

    return sum;

}
```

Console:

- For input: `80 33 27`

```
80 33 27
Sum In Function: 140
Sum In Main: 140

Process returned 0 (0x0)   execution time : 4.278 s
Press any key to continue.
```

- For input: `100 -100`

```
100 -100
Sum In Function: 0
Sum In Main: 0

Process returned 0 (0x0)   execution time : 3.699 s
Press any key to continue.
```

4. Function to calculate the sum of n numbers coming from the console and stored in an array.

Code:

```c
#include <stdio.h>

void sum(int nIntegers[], int size);

int main(int argc, char *argv[]) {

    int n;
    int summation = 0;

    scanf("%d", &n);

    int integers[n];

    for (size_t i = 0; i < n; ++i) {
        scanf("%d", &integers[i]);
    }

    sum(integers, n);

    for (size_t i = 0; i < n; i++){
        summation += integers[i];
    }

    printf("Sum In Main: %d\n", summation);

    return 0;

}

void sum(int nIntegers[], int size){

    int totalSum = 0;

    for (size_t i = 0; i < size; i++){
        totalSum += nIntegers[i];
    }
    printf("Sum In Function: %d\n", totalSum);

    return;

};
```

Console:

- For input: 3
```
  80 33 27
```

```
3
80 33 27
Sum In Function: 140
Sum In Main: 140

Process returned 0 (0x0)   execution time : 9.309 s
Press any key to continue.
```

- For input: 2
```
  100  -100
```

```
2
100 -100
Sum In Function: 0
Sum In Main: 0

Process returned 0 (0x0)   execution time : 6.001 s
Press any key to continue.
```

5. Function to swap two numbers. (Restriction: Pass by value)

Code:

```
#include <stdio.h>

void swap(int n1, int n2);

int main(int argc, char *argv[]) {
	int x, y;
	scanf("%d %d", &x, &y);
	swap(x, y);
	printf("Value in main: %d %d\n", x, y);
	return 0;
}

void swap(int n1, int n2) {
      int temp = n1;
      n1 = n2;
      n2 = temp;

      printf("Value in func: %d %d\n", n1, n2);

      return;
}
```

Console:

- For input: `10 20`

```
10 20
Value in func: 20 10
Value in main: 10 20

Process returned 0 (0x0)   execution time : 1.442 s
Press any key to continue.
```

6. Function to swap two numbers. (Restriction: Pass by reference)

Code:

```
#include <stdio.h>

void swap(int *n1, int *n2);

int main(int argc, char *argv[]) {
      int tempX, tempY, x, y, *xp, *yp;
      scanf("%d %d", &tempX, &tempY);
      x = tempX;
      y = tempY;
      xp = &tempX;
      yp = &tempY;
      swap(xp, yp);
      printf("Value in main: %d %d\n", x, y);
      return 0;
}
```

```
void swap(int *n1, int *n2) {
      int temp = *n1;
      *n1 = *n2;
      *n2 = temp;

      printf("Value in func: %d %d\n", *n1, *n2);

      return;
}
```

Console:

- For input: `10 20`

```
10 20
Value in func: 20 10
Value in main: 10 20

Process returned 0 (0x0)   execution time : 1.811 s
Press any key to continue.
```

7. Function to determine only even numbers in an array of input integers.

Code:

```
#include <stdio.h>

int extractEvens(int primary[], int *secondary, int size);

int main() {
      int n;
      scanf("%d", &n);
      int integers[n], evenIntegers[n];
      for (size_t i = 0; i < n; ++i) {
            scanf("%d", &integers[i]);
      }
      size_t j = extractEvens(integers, evenIntegers, n);
      for (size_t i = 0; i < j; ++i) {
            printf("%d", evenIntegers[i]);
            if (i != j){
                  printf(" ");
            }
      }
```

```
        return 0;
}

int extractEvens(int primary[], int *secondary, int size) {
        size_t j = 0;
        for (size_t i = 0; i < size; ++i) {
                if (primary[i]%2 == 0) {
                        secondary[j++] = primary[i];
                }
        }
        return j;
}
```

Console:

- For input: 5
  ```
  24 77 117 -512 1024
  ```
  ```
  5
  24 77 117 -512 1024
  24 -512 1024
  Process returned 0 (0x0)   execution time : 4.528 s
  Press any key to continue.
  ```

- For input: 4
  ```
  45 33 0 256
  ```
  ```
  4
  45 33 0 256
  0 256
  Process returned 0 (0x0)   execution time : 3.037 s
  Press any key to continue.
  ```

8. Function that finds and returns the minimum value in an array.

Code:

```
#include <stdio.h>

int find_minimum(int target[], int size);

int main() {
        int n;
        scanf("%d", &n);
```

```
    int integers[n];
    for (size_t i = 0; i < n; ++i) {
            scanf("%d", &integers[i]);
    }
    printf("Minimum Value: %d\n", find_minimum(integers, n));
    return 0;
}

int find_minimum(int target[], int size) {
    int minimum = target[0];
    for (int i = 1; i < size; i++) {
        if (target[i] < minimum) {
            minimum = target[i];
        }
    }
    return minimum;
}
```

Console:

- For input: 5
  ```
  157 -28 -37 26 10
  ```

  ```
  5
  157 -28 -37 26 10
  Minimum Value: -37

  Process returned 0 (0x0)   execution time : 3.167 s
  Press any key to continue.
  ```

- For input: 7
  ```
  12 45 1 10 5 3 22
  ```

  ```
  7
  12 45 1 10 5 3 22
  Minimum Value: 1

  Process returned 0 (0x0)   execution time : 3.212 s
  Press any key to continue.
  ```

9. Function that multiplies the array elements by 2 and returns the array.

   Code:

```c
#include <stdio.h>

int *multiplyBy2(int target[], int size);

int main() {
    int n;
    scanf("%d", &n);
    int integers[n];
    for (size_t i = 0; i < n; ++i) {
            scanf("%d", &integers[i]);
    }
    int *integers2x = multiplyBy2(integers, n);
      for (size_t i = 0; i < n; ++i){
            printf("%d", integers2x[i]);
            if (i != n-1) {
                  printf(" ");
            }
      }
    return 0;
}


int *multiplyBy2(int target[], int size) {
    for (size_t i = 0; i < size; ++i) {
        target[i] = target[i] * 2;
    }
    return target;
}
```

Console:

- For input: 5
```
  157 -28 -37 26 10
```
```
5
157 -28 -37 26 10
314 -56 -74 52 20
Process returned 0 (0x0)   execution time : 3.019 s
Press any key to continue.
```

- For input: 7
```
  12 45 1 10 5 3 22
```

```
7
12 45 1 10 5 3 22
24 90 2 20 10 6 44
Process returned 0 (0x0)   execution time : 3.229 s
Press any key to continue.
```

10. Function to sort and return an input array in ascending order.

Code:

```c
#include <stdio.h>

/*!
 * @brief Function for sorting
 * @details This function is a standard algorithm
 * (bubble sort) of sorting
 * that takes an array of integers and returns
 * the sorted array of integers
 *
 * @param target The target array
 * @param n The number of elements
 * @param order The order of the sorted array
 * elements. If order is ascending pass 'a', pass
 * 'd' if order is descending
 *
 * @return Sorted array
 */
int *sortArray(int target[], int n, char order);

int main(int argc, char *argv[]) {
    int n;
    scanf("%d", &n);
    int integers[n];
    for (size_t i = 0; i < n; ++i) {
        scanf("%d", &integers[i]);
    }
    int *sortedArray = sortArray(integers, n, 'a');
    for (size_t i = 0; i < n; i++) {
        printf("%d ", sortedArray[i]);
    }
    return 0;
}
```

```
/*!
 * @brief Function for sorting
 * @details This function is a standard algorithm
 * (bubble sort) of sorting
 * that takes an array of integers and returns
 * the sorted array of integers
 *
 * @param target The target array
 * @param n The number of elements
 * @param order The order of the sorted array
 * elements. If order is ascending pass 'a',
 * pass 'd' if order is descending
 *
 * @return Sorted array
 */
int *sortArray(int target[], int n, char order) {
    int i, j, temp;
    for (i = 0; i < n-1; i++) {
        for (j = 0; j < n-i-1; j++) {
            if (order == 'd' ? target[j] < target[j+1] :
target[j] > target[j+1]) {
                temp = target[j];
                target[j] = target[j+1];
                target[j+1] = temp;
            }
        }
    }
    return target;
}
```

Console:

- For input: 5
```
  10 22 -5 117 0
```

```
5
10 22 -5 117 0
-5 0 10 22 117
Process returned 0 (0x0)   execution time : 2.865 s
Press any key to continue.
```

11. Function "IsPrime()" to determine whether a number is prime or not.

Code:

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

bool IsPrime(int target);

int main(int argc, char *argv[]) {
	int x;
	scanf("%d", &x);
	if (IsPrime(x)) {
		printf("Prime");
	} else {
		printf("Not Prime");
	}

	return 0;
}

bool IsPrime(int target) {
	bool flag = true;
	if (target > 1) {
		for (size_t i = 2; i <= target/2; ++i) {
			if (target%i == 0) {
				flag = false;
			}
		}
	} else {
		flag = false;
	}

	return flag;
}
```

Console:

- For input: 1

```
1
Not Prime
Process returned 0 (0x0)   execution time : 1.128 s
Press any key to continue.
```

- For input: 2

```
2
Prime
Process returned 0 (0x0)   execution time : 1.047 s
Press any key to continue.
```

- For input: 11

```
11
Prime
Process returned 0 (0x0)   execution time : 1.126 s
Press any key to continue.
```

- For input: 39

```
39
Not Prime
Process returned 0 (0x0)   execution time : 1.236 s
Press any key to continue.
```

- For input: 101

```
101
Prime
Process returned 0 (0x0)   execution time : 1.578 s
Press any key to continue.
```

12. Function "GeneratePrime()" to compute the prime numbers less than N, where N is an input integer. GeneratePrime() uses IsPrime() to check whether a number is prime or not.

    Code:

```c
#include <stdio.h>
#include <stdbool.h>

bool IsPrime(int target);
void GeneratePrime(int term);

int main(int argc, char *argv[]) {
	int n;
	scanf("%d", &n);
	GeneratePrime(n);
	return 0;
}

bool IsPrime(int target) {
	bool flag = true;
	if (target > 1) {
		for (size_t i = 2; i <= target/2; ++i) {
			if (target%i == 0) {
				flag = false;
			}
		}
	} else {
		flag = false;
	}
	return flag;
}

void GeneratePrime(int term) {
	printf("Prime less than %d: ", term);
	for (size_t i = 0, count = 0; i < term; ++i) {
		if (IsPrime(i)) {
			printf("%s%d", count++ == 0 ? "" : ", ",i);
		}
	}
	printf("\n");
	return;
}
```

Console:

- For input: 5

```
5
Prime less than 5: 2, 3

Process returned 0 (0x0)   execution time : 1.076 s
Press any key to continue.
```

- For input: 10

```
10
Prime less than 10: 2, 3, 5, 7

Process returned 0 (0x0)   execution time : 0.888 s
```

```
Press any key to continue.
```

- For input: `40`

```
40
Prime less than 40: 2, 3, 5, 7, 11, 13, 17, 19, 23,
29, 31, 37

Process returned 0 (0x0)   execution time : 0.964 s
Press any key to continue.
```

13. Function “GenNthPrime()” to compute the N th prime number, where N is an integer input.

Code:

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

bool IsPrime(int target);
void GenNthPrime(int pos);

int main(int argc, char *argv[]) {
      int n;
      scanf("%d", &n);
      GenNthPrime(n);
      return 0;
}

bool IsPrime(int target) {
      bool flag = true;
      if (target > 1) {
            for (size_t i = 2; i <= target/2; ++i) {
                  if (target%i == 0) {
                        flag = false;
                  }
            }
      } else {
            flag = false;
      }
      return flag;
}

void GenNthPrime(int pos) {
      if (pos < 1) {
            printf("None\n");
            return;
        }
        printf("%dth Prime: ", pos);
        for (size_t i = 0, count = 1;; ++i) {
            if (IsPrime(i)) {
                if (count++ == pos) {
                    printf("%d", i);
                    break;
                }
            }
        }
        printf("\n");
        return;
    }
```

Console:

- For input: 5

```
5
5th Prime: 11

Process returned 0 (0x0)   execution time : 0.051 s
Press any key to continue.
```

- For input: 10

```
10
10th Prime: 29

Process returned 0 (0x0)   execution time : 0.606 s
Press any key to continue.
```

- For input: 40

```
40
40th Prime: 173

Process returned 0 (0x0)   execution time : 0.569 s
Press any key to continue.
```

14. Implement the following functions and calculate standard deviation of an array whose values come from the terminal –

TakeInput()
CalcMean(array, num_of_elem)
Calc_Std_deviation(array, num_of_elem)
Formula: $\sigma = \sqrt{\frac{\sum(x-M)^2}{N}}$

Code:

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

void TakeInput(int target[], int size);
double CalcMean(int target[], int size);
double Calc_Std_deviation(int target[], int size);

int main() {
    int n;
    scanf("%d", &n);
    int integers[n];
    TakeInput(integers, n);
    double std_deviation = Calc_Std_deviation(integers, n);
    printf("%0.2lf", std_deviation);
    return 0;
}

void TakeInput(int target[], int size) {
    for (int i = 0; i < size; i++) {
        scanf("%d", &target[i]);
    }
}

double CalcMean(int target[], int size) {
    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum += target[i];
    }
    return (double)sum / size;
}

double Calc_Std_deviation(int target[], int size) {
    double mean = CalcMean(target, size);
    double sum = 0;
    for (int i = 0; i < size; i++) {
```

```c
            double deviation = (double)target[i] - mean;
            sum += deviation * deviation;
        }
        return sqrt(sum / size);
    }
```

Console:

- For input: 8
  ```
  4 5 5 4 4 2 2 6
  ```

  ```
  8
  4 5 5 4 4 2 2 6
  1.32
  Process returned 0 (0x0)   execution time : 4.722 s
  Press any key to continue.
  ```

- For input: 5
  ```
  600 470 170 430 300
  ```

  ```
  5
  600 470 170 430 300
  147.32
  Process returned 0 (0x0)   execution time : 2.796 s
  Press any key to continue.
  ```

15. C Function find_substr( ) that takes two string arrays (a, b) as parameters, returns 1 if string b is found anywhere in string a, or returns -1 if no match is found. (Assuming, strlen(a) > strlen(b))

Code:

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

int find_substr(char a[], char b[]);

int main() {
    char str1[100], str2[100];
    scanf("%s", str1);
    scanf("%s", str2);
    if (find_substr(str1, str2) == 1) {
        printf("1");
    } else {
        printf("0");
```

```
    }
    return 0;
}

int find_substr(char a[], char b[]) {
    int i, j, flag;
    int len_a = strlen(a);
    int len_b = strlen(b);
    for (i = 0; i <= len_a - len_b; i++) {
        flag = 1;
        for (j = 0; j < len_b; j++) {
            if (a[i+j] != b[j]) {
                flag = 0;
                break;
            }
        }
        if (flag == 1) {
            return 1;
        }
    }
    return -1;
}
```

Console:

- For input: `madam adam`

```
madam adam
1
Process returned 0 (0x0)   execution time : 3.704 s
Press any key to continue.
```

- For input: `telescope less`

```
telescope less
0
Process returned 0 (0x0)   execution time : 3.660 s
Press any key to continue.
```

- For input: `101010 101`

```
101010 101
1
```

```
Process returned 0 (0x0)   execution time : 2.270 s
Press any key to continue.
```

16. Function find_substr( ) that takes two string arrays (a, b) as parameters, uses function str_length() to determine the lengths of the strings, and then looks for the smaller string anywhere in the bigger string. It returns 1 if the substring is found, or returns –1 if no match is found.
    [Restriction: str_length() cannot uses built-in strlen() function]

    Code:

```
#include <stdio.h>

int str_length(char str[]);
int find_substr(char a[], char b[]);

int main() {
    char str1[100], str2[100];
    scanf("%s", str1);
    scanf("%s", str2);
    if (find_substr(str1, str2) == 1) {
        printf("1");
    } else {
        printf("0");
    }
    return 0;
}

int str_length(char str[]) {
    int len = 0;
    while (str[len] != '\0') {
        len++;
    }
    return len;
}

int find_substr(char a[], char b[]) {
    int len_a = str_length(a);
    int len_b = str_length(b);
    int i, j;
    for (i = 0; i <= len_a - len_b; i++) {
        for (j = 0; j < len_b; j++) {
            if (a[i + j] != b[j]) {
                break;
            }
        }
        if (j == len_b) {
            return 1;
        }
    }
    return -1;
}
```

Console:

- For input: `madam adam`

```
madam adam
1
Process returned 0 (0x0)   execution time : 3.704 s
Press any key to continue.
```

- For input: `telescope less`

```
telescope less
0
Process returned 0 (0x0)   execution time : 3.660 s
Press any key to continue.
```

- For input: `101010 101`

```
101010 101
1
Process returned 0 (0x0)   execution time : 2.270 s
Press any key to continue.
```

17. Program that continuously takes two positive integers as inputs and uses two functions to find their GCD (greatest common divisor) and LCM (least common multiple). Both functions take parameters and returns desired values.
[Hint: Use infinite loop to process inputs]

Code:

```
#include <stdio.h>

int gcd(int a, int b);
int lcm(int a, int b);
```

```c
int main() {
    int num1, num2;
    while (1) {
        scanf("%d%d", &num1, &num2);
        int gcd_val = gcd(num1, num2);
        int lcm_val = lcm(num1, num2);
        printf("GCD: %d\n", gcd_val);
        printf("LCM: %d\n", lcm_val);
    }
    return 0;
}

int gcd(int a, int b) {
    if (b == 0) {
           return a;
    } else {
           return gcd(b, a % b);
    }
}

int lcm(int a, int b) {
    int gcd_val = gcd(a, b);
    return (a / gcd_val) * b;
}
```

Console:

- Test 1:

```
5 7
GCD: 1
LCM: 35
12 12
GCD: 12
LCM: 12
12 32
GCD: 4
LCM: 96
```

18. C Program that implements function to perform operations on a 3X5 matrix:
    InputMatrix()
    ShowMatrix()

ScalarMultiply()

Code:

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

void InputMatrix(int **matrix, int rows, int cols);
void ShowMatrix(int **matrix, int rows, int cols);
void ScalarMultiply(int **matrix, int rows, int cols, int
scalar);;

int main() {
    int rows = 3, cols = 5, scalar;

    int **matrix = (int **) malloc(rows * sizeof(int *));
    for (int i = 0; i < rows; i++) {
        matrix[i] = (int *) malloc(cols * sizeof(int));
    }

    InputMatrix(matrix, rows, cols);
    scanf("%d", &scalar);

    printf("Original:\n");
    ShowMatrix(matrix, rows, cols);

    ScalarMultiply(matrix, rows, cols, scalar);

    printf("Multiplied by %d:\n", scalar);
    ShowMatrix(matrix, rows, cols);

    for (int i = 0; i < rows; i++) {
        free(matrix[i]);
    }
    free(matrix);

    return 0;
}

void InputMatrix(int **matrix, int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }
}

void ShowMatrix(int **matrix, int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d\t", matrix[i][j]);
        }
        printf("\n");
    }
}

void ScalarMultiply(int **matrix, int rows, int cols, int
scalar) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            matrix[i][j] *= scalar;
        }
    }
}
```

Console:

- Test 1:

```
7 16 55 13 12
12 10 52 0 7
-2 1 2 4 9
2
Original:
7       16      55      13      12
12      10      52      0       7
-2      1       2       4       9
Multiplied by 2:
14      32      110     26      24
24      20      104     0       14
-4      2       4       8       18

Process returned 0 (0x0)   execution time : 2.775 s
Press any key to continue.
```

- Test 2:

```
7 16 55 13 12
12 10 52 0 7
-2 1 2 4 9
-1
Original:
7       16      55      13      12
12      10      52      0       7
-2      1       2       4       9
Multiplied by -1:
-7      -16     -55     -13     -12
-12     -10     -52     0       -7
2       -1      -2      -4      -9

Process returned 0 (0x0)   execution time : 4.435 s
Press any key to continue.
```

19. C Program that implements function to perform operations on a 3X5 matrix:
    InputMatrix()
    ShowMatrix()
    ScalarMultiply()

Code:

```c
#include <stdio.h>
#include <stdlib.h>

void InputMatrix(int **matrix, int rows, int cols);
void ShowMatrix(int **matrix, int rows, int cols);
void ScalarMultiply(int **matrix, int rows, int cols, int
scalar);;

int main() {
    int rows, cols, scalar;

    scanf("%d %d", &rows, &cols);

    int **matrix = (int **) malloc(rows * sizeof(int *));
    for (int i = 0; i < rows; i++) {
        matrix[i] = (int *) malloc(cols * sizeof(int));
    }

    InputMatrix(matrix, rows, cols);
    scanf("%d", &scalar);
    printf("Original:\n");
    ShowMatrix(matrix, rows, cols);

    ScalarMultiply(matrix, rows, cols, scalar);

    printf("Multiplied by %d:\n", scalar);
    ShowMatrix(matrix, rows, cols);

    for (int i = 0; i < rows; i++) {
        free(matrix[i]);
    }
    free(matrix);

    return 0;
}

void InputMatrix(int **matrix, int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }
}

void ShowMatrix(int **matrix, int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d\t", matrix[i][j]);
        }
        printf("\n");
    }
}

void ScalarMultiply(int **matrix, int rows, int cols, int
scalar) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            matrix[i][j] *= scalar;
        }
    }
}
```

Console:

- Test 1:

```
2 2

7 16
12 10

2
Original:
7       16
12      10
Multiplied by 2:
14      32
24      20

Process returned 0 (0x0)   execution time : 12.522 s
Press any key to continue.
```

- Test 2:

```
3 5

7 16 55 13 12
12 10 52 0 7
-2 1 2 4 9

-1
Original:
7       16      55      13      12
12      10      52      0       7
-2      1       2       4       9
Multiplied by -1:
-7      -16     -55     -13     -12
-12     -10     -52     0       -7
2       -1      -2      -4      -9

Process returned 0 (0x0)   execution time : 30.833 s
Press any key to continue.
```

20. C Program to convert a positive integer to another base using the following functions-
    I. Get_Number_And_Base () : Takes number to be converted (N) and base value (B) from user. Base must be between 2 and 16.

II. Convert_Number () : Does the conversion
III. Show_Converted_Number() : Displays the converted value.

Code:

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

bool Get_Number_And_Base(int *n, int *b);
void Convert_Number(int n, int b, char *result);
void Show_Converted_Number(int n, int b);

int main(int argc, char *argv[]) {
    int n, b;
    if (Get_Number_And_Base(&n, &b)) {
            Show_Converted_Number(n, b);
    }
    return 0;
}

bool Get_Number_And_Base(int *n, int *b) {
    scanf("%d", n);
    scanf("%d", b);
    if (*b < 2 || *b > 16) {
        printf("Base not within proper range!\n");
        return false;
    } else {
            return true;
    }
}

void Convert_Number(int n, int b, char *result) {
    int i = 0;
    while (n != 0) {
        int rem = n % b;
        if (rem < 10) {
            result[i] = rem + '0';
        } else {
            result[i] = rem - 10 + 'A';
        }
        i++;
        n /= b;
    }
    result[i] = '\0';
    return;
}

void Show_Converted_Number(int n, int b) {
    char result[100];
    Convert_Number(n, b, result);

    int len = strlen(result);

    for (int i = 0; i < len / 2; i++) {
        char temp = result[i];
        result[i] = result[len - i - 1];
        result[len - i - 1] = temp;
    }

    printf("%s\n", result);
    return;
}
```

Console:

- For input: `100 8`

```
100 8
144

Process returned 0 (0x0)   execution time : 7.077 s
Press any key to continue.
```

- For input: `512 16`

```
512 16
200

Process returned 0 (0x0)   execution time : 1.956 s
Press any key to continue.
```

- For input: `512 0`

```
512 0
```

```
Base not within proper range!

Process returned 0 (0x0)   execution time : 1.860 s
Press any key to continue.
```

THE END