To:
Tenouk

C PROGRAMMING:
THE IF, WHILE, DO-WHILE, FOR AND ARRAY
WORKING PROGRAM EXAMPLES (with some flowcharts)

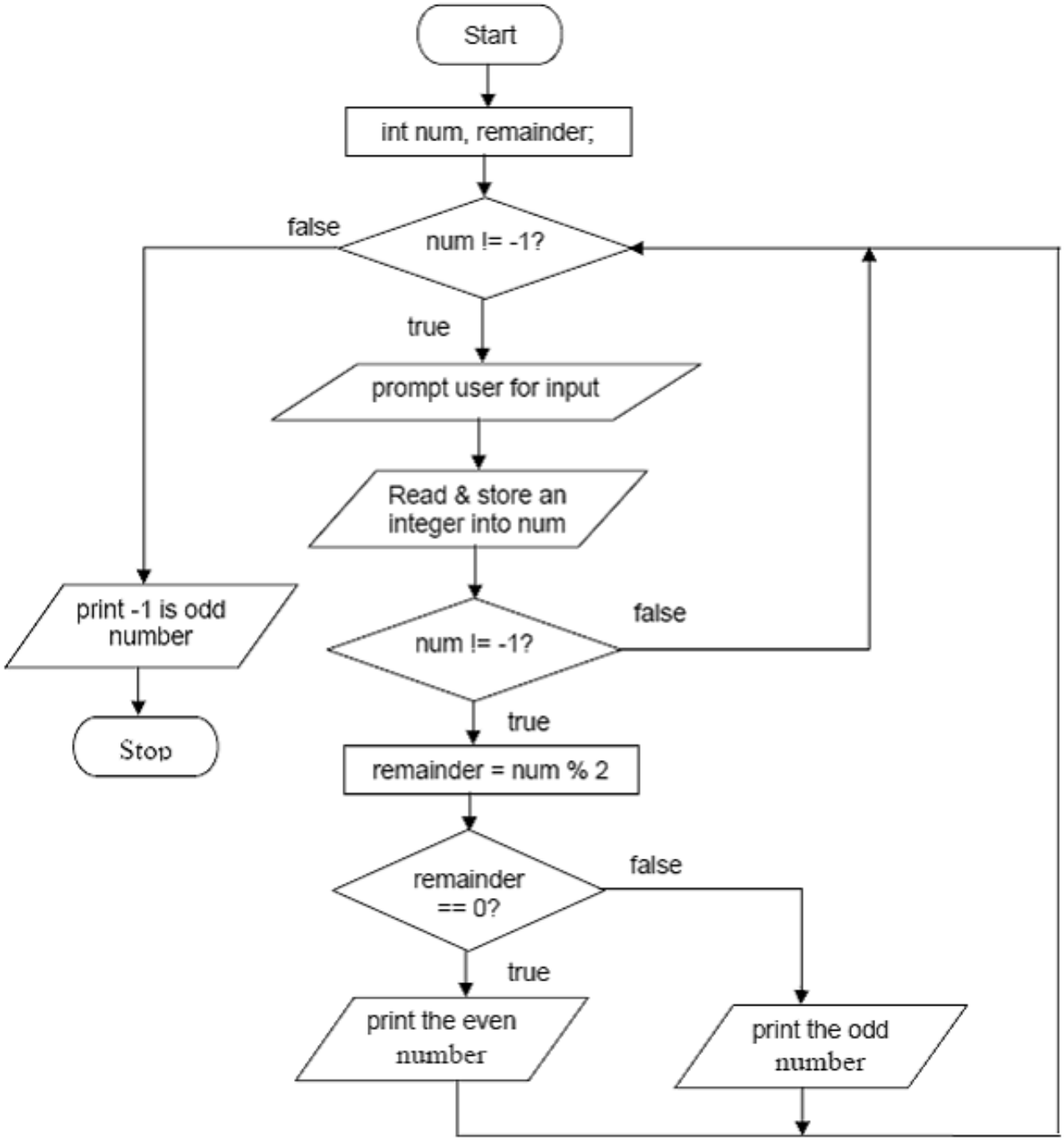
1. Compiler: VC++ Express Edition 2005
2. Project: Win32 > Win32 Console Application
3. Setting: No Common Language Runtime support, Use Unicode Character Set and Compile as C Code (/TC) (others are default).
4. OS: Win Xp Pro SP2 + updates + patches…, 2GB RAM, Intel Core 2 Duo…

1. Write a program that reads an integer and checks whether it is odd or even. For example:

Enter a number: 25
25 is an odd number.

Answer:

The following is an algorithm for this program using a flow chart. We can use a modulus operator to solve this problem. There will be no remainder for even number when we modulus the number by 2.

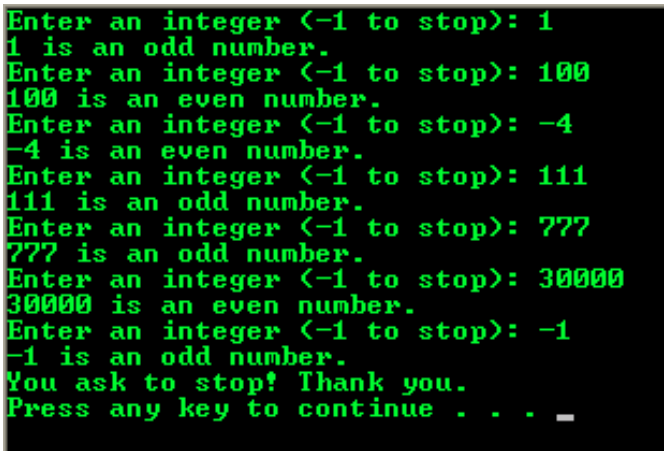


The source code:

```
#include <stdio.h>

int main()
{
	int num = 0, remainder = 0;

	// while -1 not entered...
	while(num != -1)
	{
		// prompt user for input
		printf("Enter an integer (-1 to stop): ");
		// read and store input, then modulus by 2
		scanf_s("%d", &num, sizeof(int));
		// ready to stop if -1 else...
		if(num != -1)
		{
			remainder = num % 2;
```

```
                // test for even/odd. If the modulus yields 0, it is even
                if(remainder == 0)
                        printf("%d is an even number.\n", num);
                else
                        printf("%d is an odd number.\n", num);
        }
    }
    // -1 was entered
    printf("%d is an odd number.\n", num);
    printf("You ask to stop! Thank you.\n");
    return 0;
}
```

A sample output:



The do-while version.

```
#include <stdio.h>

int main()
{
    int num = 0, remainder = 0;

    do
    {
        // prompt user for input
        printf("Enter an integer (-1 to stop): ");
        // read and store input, then modulus by 2
        scanf_s("%d", &num, sizeof(int));
        // ready to stop if -1 else...
        if(num != -1)
        {
            remainder = num % 2;
            // test for even/odd. If the modulus yields 0, it is even
            if(remainder == 0)
                printf("%d is an even number.\n", num);
            else
                printf("%d is an odd number.\n", num);
        }
    }// while -1 not entered...
    while(num != -1);
    // -1 was entered
    printf("%d is an odd number.\n", num);
    printf("You ask to stop! Thank you.\n");
```

```
    return 0;
}
```

2. The *wind chill index* (WCI) is calculated from the wind speed v in miles per hour and the temperature t in Fahrenheit. Three formulas are used, depending on the wind speed:

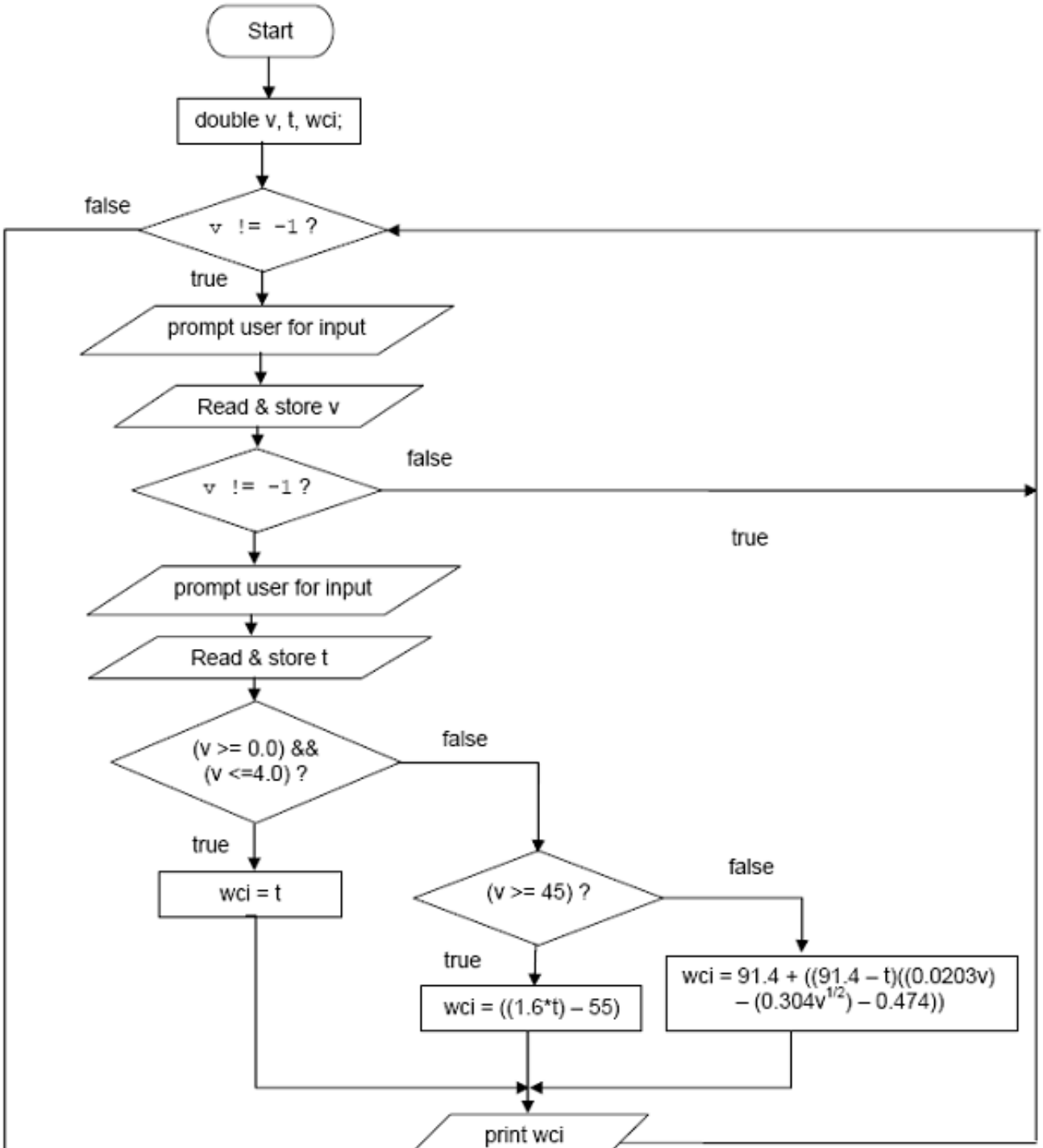
if (0 <= v <= 4) then WCI = t

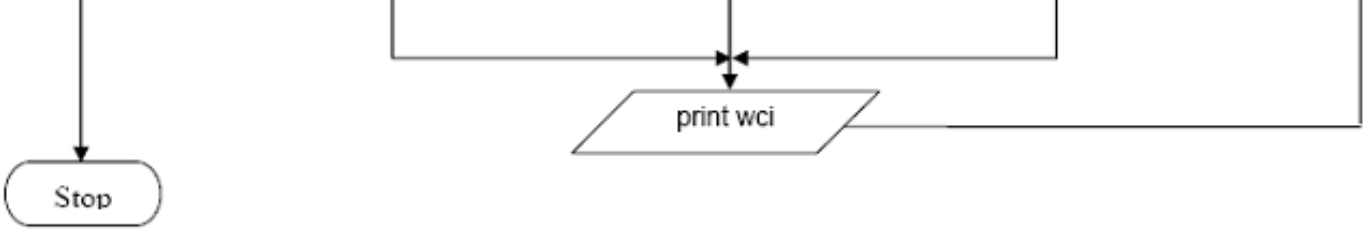
if (v >=45) then WCI = 1.6t - 55

otherwise, WCI = 91.4 + (91.4 - t)(0.0203v - 0.304$(v)^{1/2}$ - 0.474). Write a program that can calculate the wind chill index.

Answer:

The if-else is suitable for this solution, choosing from three conditional expressions. We need to prompt user for *v* and *t* in order to calculate and show the *wci*. The following is an algorithm for this program using a flow chart.



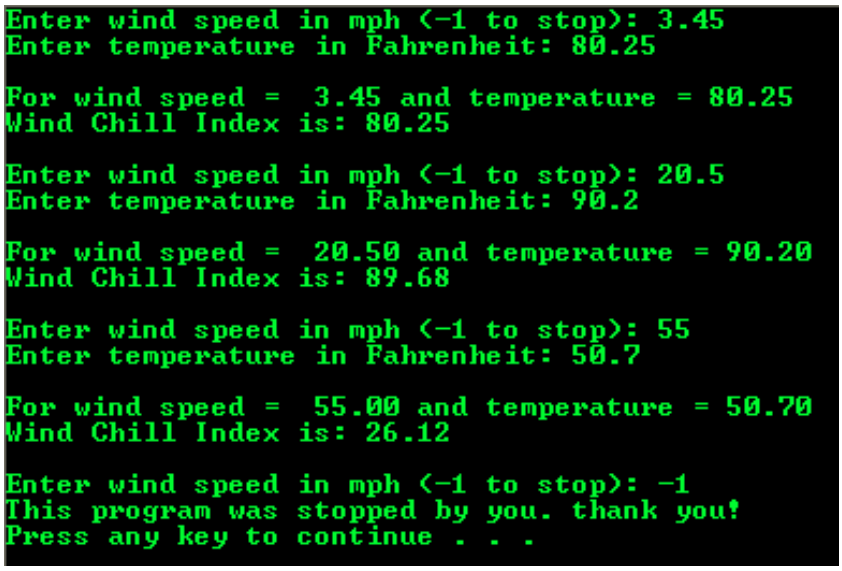


The source code:

```
#include <stdio.h>
// for pow(x,y)
#include <math.h>

int main()
{
	// v is wind speed in mph, t is temperature in Fahrenheit
	// and wci is wind chill index
	double v = 0.0, t = 0.0, wci = 0.0;

	// let provide a loop for continuous input until stopped by user
	while(v != -1)
	{
		// read and store v from user inputs
		printf("Enter wind speed in mph (-1 to stop): ");
		// the 3rd parameter of scanf_s() is not required for numerical,
int and float
		// the lf is for double or long int, the l (el) is
microsoft extension...
		scanf_s("%lf", &v, sizeof(double));
		// if user don't want to stop then repeat...
		if(v != -1)
		{
			// read and store t from user inputs
			// the 3rd parameter of scanf_s() is not required
for numerical, int and float
			printf("Enter temperature in Fahrenheit: ");
			scanf_s("%lf", &t, sizeof(double));
			// if (0 <= v <= 4)
			if((v >= 0.0) && (v <=4.0))
				wci = t;
			// if (v >= 45)
			else if (v >= 45)
				wci = ((1.6*t) - 55);
			// others...
			else
				wci = 91.4 + ((91.4 - t)*((0.0203*v) - (0.304*
(pow(v,0.5))) - 0.474));
			// print one of the result
			printf("\nFor wind speed =  %.2f and temperature = %.2f\n",
v, t);
			printf("Wind Chill Index is: %.2f\n", wci);
			printf("\n");
		}
		// check the while loop condition
	}
	// if user press -1 for wind speed then stop...
	printf("This program was stopped by you. thank you!\n");
```

```
    return 0;
}
```

A sample output:

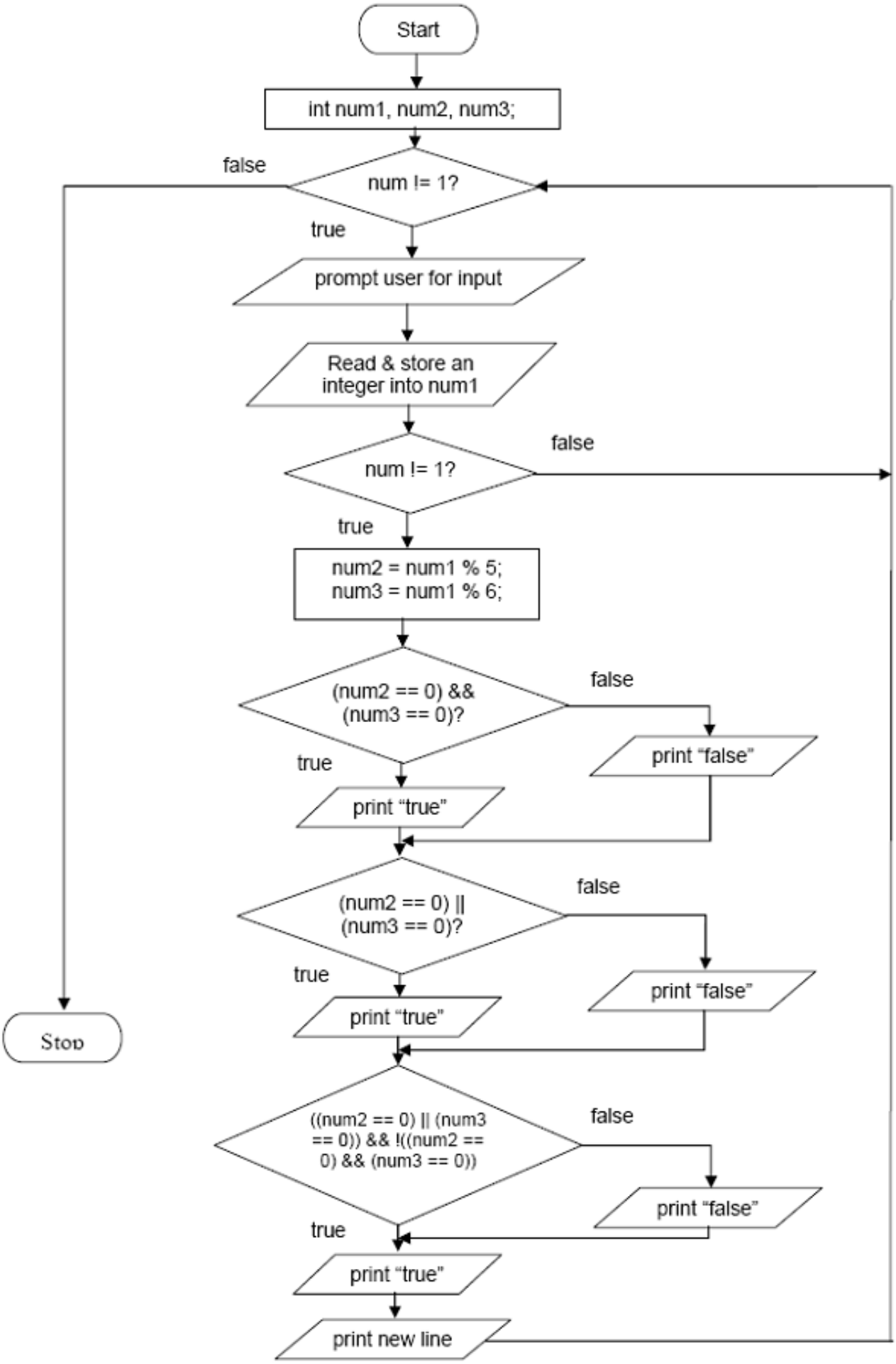


3. Write a program that asks the user to enter an integer and determines whether it is divisible by 5 and 6, whether it is divisible by 5 or 6, and whether it is divisible by 5 or 6 but not both. For example, if your input is 10, the output should be:

   Is 10 divisible by 5 and 6? false
   Is 10 divisible by 5 or 6? true
   Is 10 divisible by 5 or 6, but not both? true

   Answer:

   We can use the logical AND (&&), OR (||), NOT (!) and modulus (%) to solve this problem. If the modulus yields a 0, the number is divisible otherwise it is not divisible. Then we use the logical operators to provide the desired outputs. The following is an algorithm for this program using a flow chart.



The source code:

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

```c
int main()
{
	int num1 = 0, num2 = 0, num3 = 0;

	while(num1 != -1)
	{
		// read and store an integer from user
		printf("Enter an integer, -1 to stop: ");
		scanf_s("%d", &num1);
		// check whether user want to stop or not
		if(num1 != -1)
		{
			// Let determine the divisibility of 5 and 6
			num2 = num1 % 5; // num2 = 0, divisible
			num3 = num1 % 6; // num3 = 0, divisible

			// in this example, all three conditions must be tested
			// do the equality comparison
			// Divisible by 5 AND 6?
			if((num2 == 0) && (num3 == 0))
				printf("Is %d divisible by 5 and 6? true\n", num1);
			else
				printf("Is %d divisible by 5 and 6? false\n", num1);

			// Divisible by 5 OR 6?
			if((num2 == 0) ||(num3 == 0))
				printf("Is %d divisible by 5 or 6? true\n", num1);
			else
				printf("Is %d divisible by 5 or 6? false\n", num1);

			// Divisible by 5 OR 6 but NOT both?
			if(((num2 == 0) ||(num3 == 0)) && !((num2 == 0) && (num3 == 0)))
				printf("Is %d divisible by 5 or 6 but not both? true
\n", num1);
			else
				printf("Is %d divisible by 5 or 6 but not both? false
\n", num1);
		}
		printf("\n");
		// check the while condition
	}
	// exit message
	printf("You asked to stop. Thank you!\n");
	return 0;
}
```

A sample output:

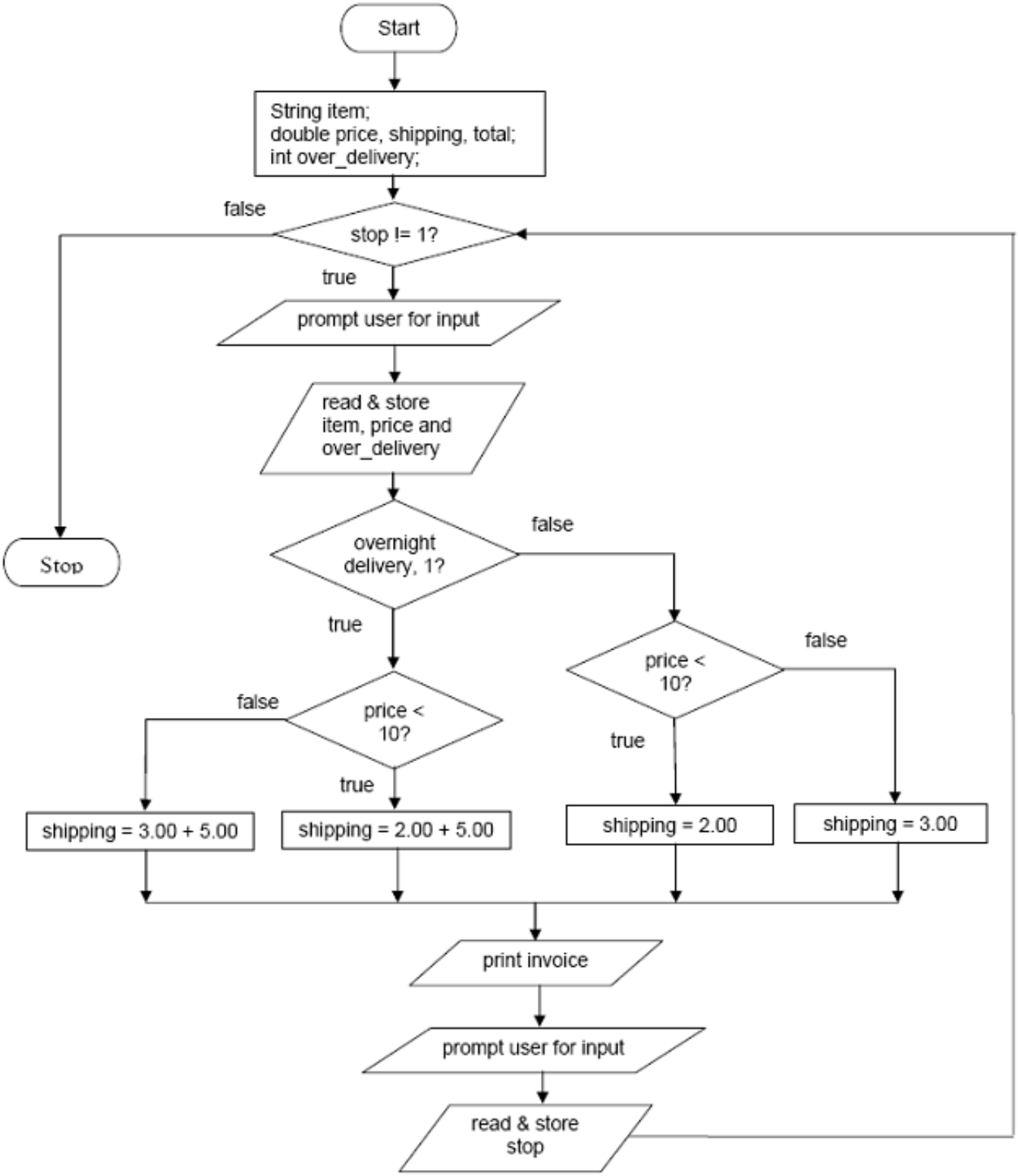
```
Enter an integer, -1 to stop: 5
Is 5 divisible by 5 and 6? false
Is 5 divisible by 5 or 6? true
Is 5 divisible by 5 or 6 but not both? true

Enter an integer, -1 to stop: 6
Is 6 divisible by 5 and 6? false
Is 6 divisible by 5 or 6? true
Is 6 divisible by 5 or 6 but not both? true

Enter an integer, -1 to stop: 1
Is 1 divisible by 5 and 6? false
Is 1 divisible by 5 or 6? false
Is 1 divisible by 5 or 6 but not both? false

Enter an integer, -1 to stop: 30
Is 30 divisible by 5 and 6? true
Is 30 divisible by 5 or 6? true
Is 30 divisible by 5 or 6 but not both? false

Enter an integer, -1 to stop: -30
Is -30 divisible by 5 and 6? true
Is -30 divisible by 5 or 6? true
Is -30 divisible by 5 or 6 but not both? false

Enter an integer, -1 to stop: -1

You asked to stop. Thank you!
Press any key to continue . . .
```

4. MyJava Café wants you to write a program to take orders from the Internet. Your program asks for the item, its price, and if overnight shipping is wanted. Regular shipping for items under $10 is $2.00; for items $10 or more shipping is $3.00. For overnight delivery add $5.00. For example, the output might be:

```
Enter the item:
Tuna Salad
Enter the price:
450
Overnight delivery (0==no, 1==yes):
1

Invoice:
  Tuna Salad    4.50
  shipping      7.00
  total        11.50
```

Answer:

Using the nested if-else, we test the overnight delivery condition that chosen by user. After confirming the overnight delivery, on the true path, we test the amount of price whether it is less than $10 or not. On the false side, we also test the price whether less than $10 or not and finally print the total price for the respective condition. The following is an algorithm for this program using a flow chart.

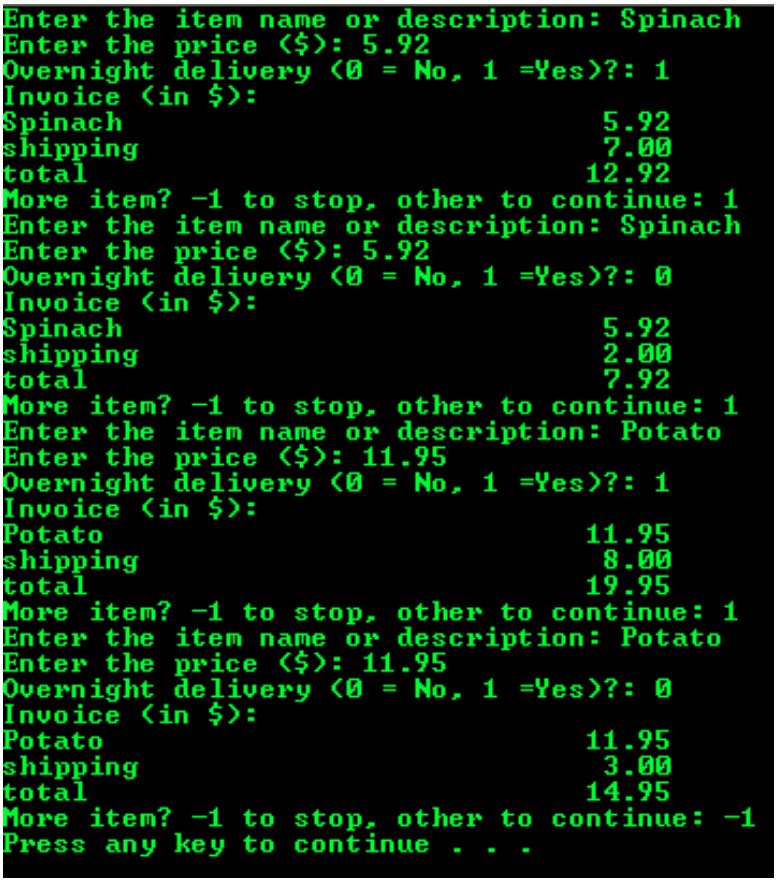


The source code:

```
#include <stdio.h>
// for strcmp()
#include <string.h>

int main()
{
	char item[20]= "";
	double price = 0.0, shipping = 0.0, total = 0.0;
	int over_delivery, stop = 1;

	while(stop != -1)
	{	// if stop != 1, continue...
```

```
            // prompt for user input
            printf("Enter the item name or description: ");
            // the 3rd parameter is required for character and string
            // store item
            scanf_s("%s", item, sizeof(item));
            // prompt user for price
            printf("Enter the price ($): ");
            // store price
            scanf_s("%lf", &price);
            // prompt user for overnight delivery choice
            printf("Overnight delivery (0 = No, 1 =Yes)?: ");
            // store the choice
            scanf_s("%d", &over_delivery);

            // if the overnight delivery is needed...
            if(over_delivery == 1)
            {
                  if(price < 10)
                        shipping = 2.00 + 5.00;
                  else
                        shipping = 3.00 + 5.00;
            }
            // if no overnight delivery
            if (over_delivery == 0)
            {
                  if(price < 10)
                        shipping = 2.00;
                  else
                        shipping = 3.00;
            }
            // print all the results
            printf("Invoice (in $):\n");
            printf("%-23s %15.2f\n", item, price);
            printf("shipping %30.2f\n", shipping);
            total = price + shipping;
            printf("total %33.2f\n", total);
            // prompt user for continuation....
            printf("More item? -1 to stop, other to continue: ");
            scanf_s("%d", &stop, sizeof(int));
      }
      return 0;
}
```

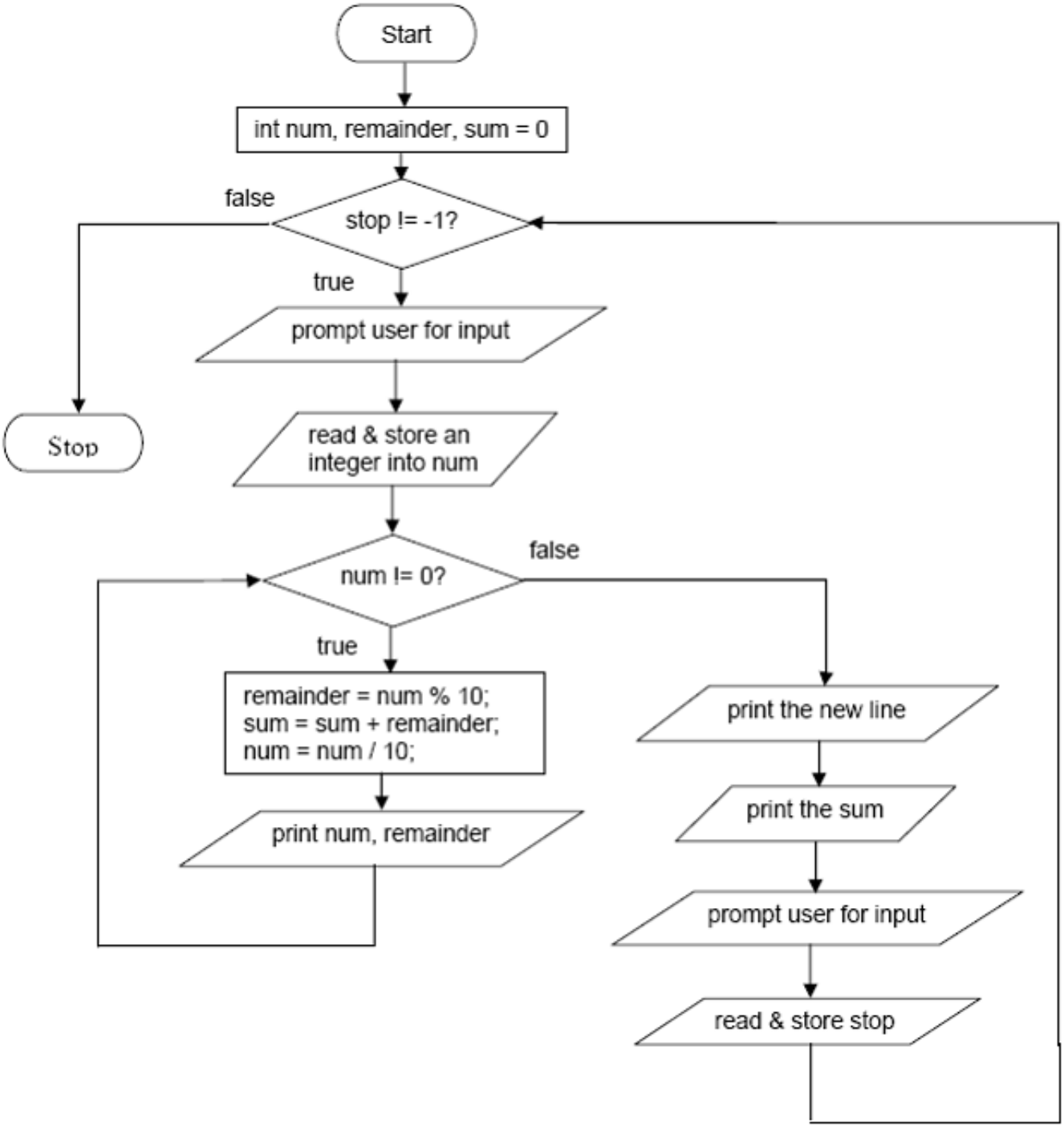
A sample output:



5. Write a program that reads an integer between 0 – 999 and adds all the digits in the integer. For example, if an integer is 932, the sum of all its digit is 14. Hint: Use the % operator to extract digits and use the / operator to remove the extracted digit. For instance, 932 % 10 = 2 and 932 / 10 = 93.

   Answer:

   The sum of integer digits is the sum of the remainder when the integer is repeatedly modulus'ed by 10 and then divided by 10 until the integer becomes 0. For repetition we can use the while loop. The following is an algorithm for this program using a flow chart.



The source code:

```
#include <stdio.h>

int main()
{
	int count = 0, num = 0, remainder = 0, sum = 0, stop = 0;

	while(stop != -1)
	{
		printf("Enter an integer: ");
		scanf_s("%d", &num);
		// test the num == 0?
		printf("\nAfter operation:\n");
		printf("remainder num\n");
		printf("--------- ---\n");
		while(num != 0)
```

```
            {
                  // get the remainder (digits) by dividing by 10
                  remainder = num % 10;
                  // sum up the remainder
                  sum = sum + remainder;
                  // divide the number by 10, next integer part
                  // ...10000, 1000, 100, 10, 0
                  num = num / 10;
                  // let see current value of num and remainder...
                  printf("%d        %d\n", remainder, num);
            }
            printf("\n");
            // print the sum of the digits...
            printf("The sum of digits is %d\n", sum);
            // reset sum to 0, for next test
            sum = 0;
            printf("More? -1 to stop, other to continue: ");
            scanf_s("%d",&stop);
      }
      return 0;
}
```

A sample output:

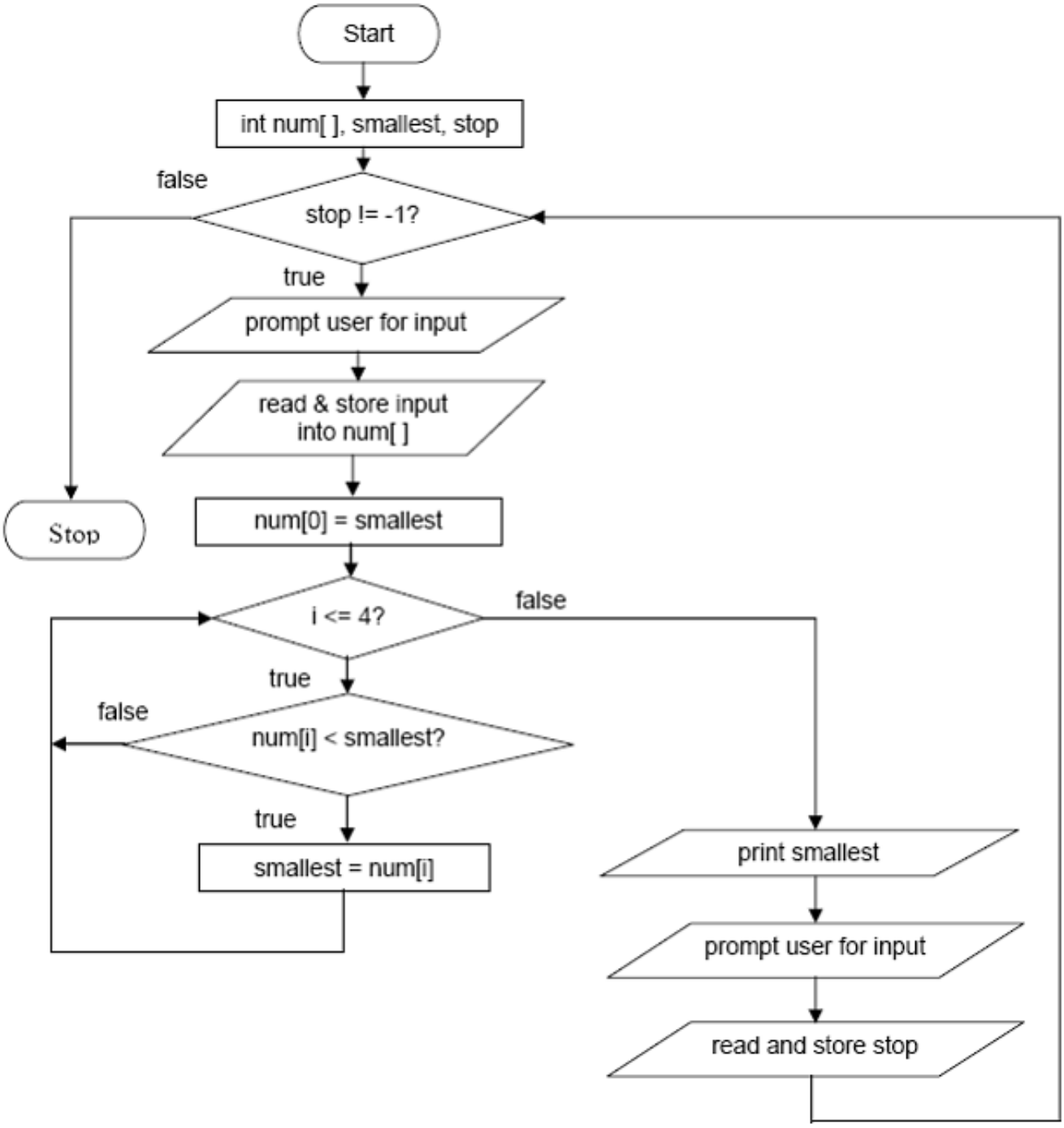
```
Enter an integer: 1234

After operation:
remainder       num
---------       ---
4               123
3               12
2               1
1               0

The sum of digits is 10
More? -1 to stop, other to continue: 0
Enter an integer: 50001

After operation:
remainder       num
---------       ---
1               5000
0               500
0               50
0               5
5               0

The sum of digits is 6
More? -1 to stop, other to continue: 0
Enter an integer: 1111

After operation:
remainder       num
---------       ---
1               111
1               11
1               1
1               0

The sum of digits is 4
More? -1 to stop, other to continue: 0
Enter an integer: 90000

After operation:
remainder       num
---------       ---
0               9000
0               900
0               90
0               9
9               0

The sum of digits is 9
More? -1 to stop, other to continue: -1
Press any key to continue . . .
```

```
9                0

The sum of digits is 9
More? -1 to stop, other to continue: -1
Press any key to continue . . .
```

6. Write a program that can read three integers from the user and then determines the smallest value among the three integers.

   Answer:

   The using of if statement is not the efficient way for the solution. It is better to use an array with loop, mainly when there is a list of integer. The following is an algorithm for this program using a flow chart.



The source code:

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

```
int main()
{
	int i, num[5], smallest = 0, stop = 0;

	while( stop != -1)
	{
		// prompt input from user
		printf("Enter 5 integers separated by a space: ");
		// store those integers in an array
		for(i=0;i <=4;i++)
			scanf_s("%d", &num[i]);

		// assign the 1st element to smallest
		smallest = num[0];
		// compare the others and keep storing the smallest
		for(i=1;i<=4;i++)
			if(num[i] < smallest)
				smallest = num[i];
		// print some text...
		printf("The smallest number among ");
		// print the element
		for(i=0;i <=4;i++)
			printf("%d ", num[i]);
		// print the smallest
		printf("is %d\n", smallest);
		printf("\nMore data? -1 to stop, others to continue: ");
		scanf_s("%d", &stop);
	}
	return 0;
}
```

A sample output:

```
Enter 5 integers separated by a space: 10 2 12 7 5
The smallest number among 10 2 12 7 5 is 2

More data? -1 to stop, others to continue: 1
Enter 5 integers separated by a space: 30 21 45 15 10
The smallest number among 30 21 45 15 10 is 10

More data? -1 to stop, others to continue: 1
Enter 5 integers separated by a space: 15 10 7 8 11
The smallest number among 15 10 7 8 11 is 7

More data? -1 to stop, others to continue: 1
Enter 5 integers separated by a space: 7 2 3 8 0
The smallest number among 7 2 3 8 0 is 0

More data? -1 to stop, others to continue: -1
Press any key to continue . . .
```

By changing the if statement:

```
if(num[i] < smallest)
	smallest = num[i];
```

To

```
if(num[i] > smallest)
	smallest = num[i];
```

Will scan the largest number as shown in the following example.

```
#include <stdio.h>

int main()
{
	int i, num[5], largest = 0, stop = 0;

	while( stop != -1)
	{
		// prompt input from user
		printf("Enter 5 integers separated by a space: ");
		// store those integers in an array
		for(i=0;i <=4;i++)
			scanf_s("%d", &num[i]);

		// assign the 1st element to largest
		largest = num[0];
		// compare the others and keep storing the largest
		for(i=1;i<=4;i++)
			if(num[i] > largest)
				largest = num[i];
		// print some text...
		printf("The largest number among ");
		// print the element
		for(i=0;i <=4;i++)
			printf("%d ", num[i]);
		// print the largest
		printf("is %d\n", largest);
		printf("\nMore data? -1 to stop, others to continue: ");
		scanf_s("%d", &stop);
	}
	return 0;
}
```

A Sample output:

```
More data? -1 to stop, others to continue: 1
Enter 5 integers: 12 3 14 20 71
The largest number among 12 3 14 20 71 is 71

More data? -1 to stop, others to continue: 1
Enter 5 integers: 19 0 -8 21 8
The largest number among 19 0 -8 21 8 is 21

More data? -1 to stop, others to continue: -1
Press any key to continue . . .
```

7. Write a program that asks the user to input an integer and then outputs the individual digits of the number.

Answer: Using the division (/) and modulus (%).

```
#include <stdio.h>
// for pow(x,y)
#include <math.h>

int main()
```

```
{
   //------------------------------------------------------------------------
   // Separating an integer to individual digits
   // The x % y computes the remainder obtained when x is divided by y.
   //------------------------------------------------------------------------

      // can try long for bigger range, int range is the limit
      int intnumber = 0, condition = 0, remainder = 0;
      // counter to store the number of digit entered by user
      int counter = 0;

      // prompt user for input
      printf("Enter an integer number: ");
      // read and store input in intnumber
      scanf_s("%d", &intnumber);
      // set the condition sentinel value to intnumber
      condition = intnumber;

      // we need to determine the number of digit
      // entered by user, we don't know this and store it in counter
      while (condition > 0)
      {
         condition  = condition / 10;
         counter = counter + 1;
      }

      // well, we already know the number of digit entered by user,
      // start with number of digits less 1, because we need to discard
      // the last one...
      counter = counter - 1;
      printf("The individual digits: ");
      while (counter >= 0)
      {
         // extract each of the decimal digits, need to cast to int
         // to discard the fraction part
         // pow(10, counter) used to determine the ...,10000, 1000, 100, 10, 1
         // because initially we don't know how many digits user entered...
         remainder = intnumber % (int) pow(10, counter);
         intnumber = intnumber/(int) pow(10,counter);
         printf("%d  ", intnumber);
         intnumber = remainder;
         counter = counter - 1;
      }
        printf("\n");
   return 0;
}
```

A sample output:

```
Enter an integer number: 780012
The individual digits: 7  8  0  0  1  2
Press any key to continue . . .
```

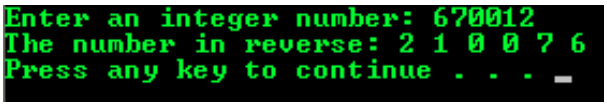
8. Write a program that asks the user to input an integer and then outputs the number with the digits reversed.

Answer: This is a previous answer with an array to store the integer digits and then read the array reversely.

```c
#include <stdio.h>
// for pow(x,y)
#include <math.h>

int main()
{

      // can try long for bigger range, int range is the limit
      int intnumber, condition, remainder;
      // counter to store the number of digit entered by user
      // counter1 is similar, used as for loop sentinel
      int counter = 0, i = 0, j = 0, counter1 = 0;
      int reverseint[20];

      // prompt user for input
      printf("Enter an integer number: ");
      // read and store input in intnumber
      scanf_s("%d", &intnumber);
      // set the condition sentinel value to intnumber
      condition = intnumber;

      // we need to determine the number of digit
      // entered by user and store it in counter
      while (condition > 0)
      {
         condition  = condition /10;
         counter = counter + 1;
         // this counter for printing in reverse
         counter1 = counter1 + 1;
      }

      // well, we already know the number of digit entered by user,
      // start with number of digits less 1, because we need to discard
      // the last one, pow(10,1)
      counter = counter - 1;
      printf("The number in reverse: ");
      while (counter >= 0)
     {
         // extract each of the decimal digits, need to cast to int
         // to discard the fraction part
         // pow(10, counter) used to determine the ...,10000, 1000, 100, 10, 1
         // because initially we don't know how many digits user entered...
         remainder = intnumber % (int) pow(10, counter);
         intnumber = intnumber/(int) pow(10,counter);

         // store the digits in an array for later use
         reverseint[i] = intnumber;
         i++;
         // update and repeat for the rest
         intnumber = remainder;
         counter = counter - 1;
      }
      // print the array element in reverse
      for(j=counter1-1;j >= 0;j--)
              printf("%d ", reverseint[j]);
        printf("\n");
   return 0;
```

```
}
```

A sample output:



9. Write a program that asks the user to enter any number of integers that are in the range of 0 to 30 inclusive and count how many occurrences of each number are entered. Use a suitable sentinel to signal the end of input. Print out only the numbers (with the number of occurrences) that were entered one or more times. (Note: You must use array in your solution for this problem)

Answer: Uses 3 arrays, one for storing the input, one used for comparison to count the occurrences and another on to store the count of occurrences. Display the content of the third array.

```
#include <stdio.h>
// for pow(x,y)
#include <math.h>

int main()
{
     // used to store the input by user
      int myint[50];
      // used to compare myint[] to every element for occurrences
      int mycompare[31];
      // used to store the count of occurrences, initially all element default
to 0
        // to make sure there is no 'rubbish' stored for the number of input that
        // less than 31...array with content of 0 will be used later
        int mycount[31] = {0};
      // array indexes
      int i = 0, j = 0, k = 0, sum = 0;

      // fill in mycompare[] for comparison
      for(j=0;j <= 30;j++)
         mycompare[j] = j;

     // prompt user for input until stopped
      do
      {
         printf("Enter integer between 0 and 30 inclusive, other to stop: ");
         // store user input in myint[]
         scanf_s("%d", &myint[i]);

         // do a comparison
         for(j=0;j<=30;j++)
            for(k=0;k<=30;k++)
            {
               // make sure the index is same
               j = k;
               // compare the user input to every mycompare[] values
               // if similar….
               if(myint[i] == mycompare[j])
                  // ...if similar, store the count at similar index of
mycompare[]
                  mycount[k] = mycount[k]+1;
            }
```

```
        // increase counter for next input
        i++;
      // the sentinel range values, minus 1 for the last user input
     }while((myint[i-1] >=0) && (myint[i-1] <= 30));
     // print the results that already stored in mycount[]
     printf("Number\t\tCount\n");
     printf("======\t\t=====\n");
     // iterate all element…
     for(k=0; k <=30 ;k++)
     {
        // ..but, just print the number that having count
        if(mycount[k] != 0)
        {
              printf("%d\t\t%d\n",k, mycount[k]);
              sum = sum + mycount[k];
        }
     }
     printf("Total user input = %d\n", sum);

  return 0;
}
```

A sample output:

```
Enter integer between 0 and 30 inclusive, other to stop: 3
Enter integer between 0 and 30 inclusive, other to stop: 5
Enter integer between 0 and 30 inclusive, other to stop: 6
Enter integer between 0 and 30 inclusive, other to stop: 7
Enter integer between 0 and 30 inclusive, other to stop: 12
Enter integer between 0 and 30 inclusive, other to stop: 2
Enter integer between 0 and 30 inclusive, other to stop: 3
Enter integer between 0 and 30 inclusive, other to stop: 7
Enter integer between 0 and 30 inclusive, other to stop: 6
Enter integer between 0 and 30 inclusive, other to stop: 5
Enter integer between 0 and 30 inclusive, other to stop: 6
Enter integer between 0 and 30 inclusive, other to stop: 4
Enter integer between 0 and 30 inclusive, other to stop: 10
Enter integer between 0 and 30 inclusive, other to stop: 3
Enter integer between 0 and 30 inclusive, other to stop: 8
Enter integer between 0 and 30 inclusive, other to stop: 9
Enter integer between 0 and 30 inclusive, other to stop: 7
Enter integer between 0 and 30 inclusive, other to stop: 5
Enter integer between 0 and 30 inclusive, other to stop: 3
Enter integer between 0 and 30 inclusive, other to stop: -1
Number          Count
======          =====
2               1
3               4
4               1
5               3
6               3
7               3
8               1
9               1
10              1
12              1
Total user input = 19
Press any key to continue . . .
```

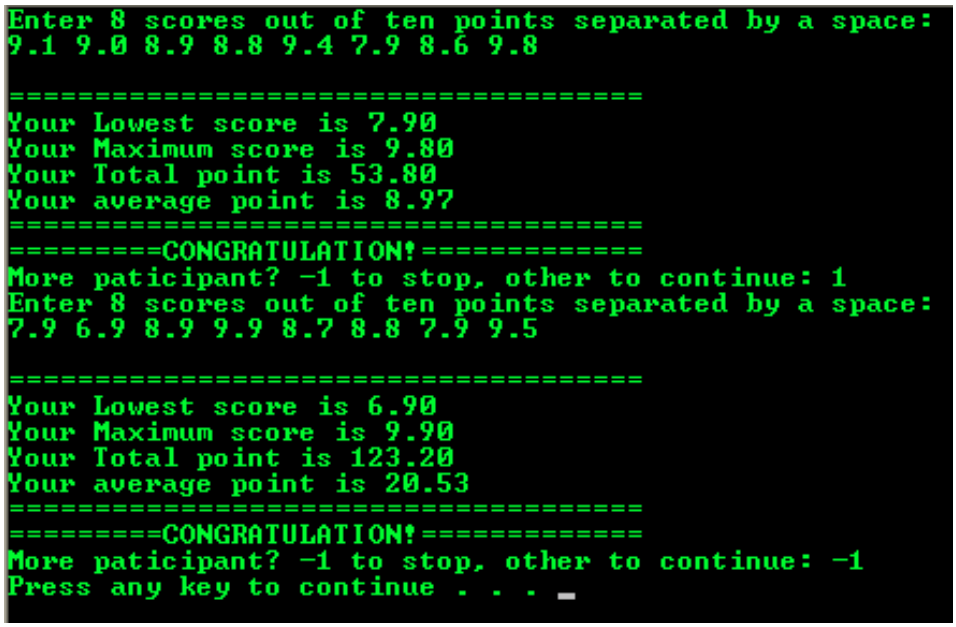
10. In a gymnastics or diving competition, each contestant's score is calculated by dropping the lowest and highest scores and then adding the remaining scores. Write a program that allows the user to enter eight judges' scores and then outputs the point received by the contestant. A judge awards point between 1 and 10, with 1 being the lowest and 10 being the highest. For example, if the scores are: 9.2, 9.3, 9.0, 9.9, 9.5, 9.5, 9.6 and 9.8, then the contestant receives a total of 56.9 points. (Note: You must use array in your solution for this problem)

Answer: Store the result in an array and then manipulate the array elements.

```
#include <stdio.h>

int main()
{
	double maxScore = 0.0,  minScore = 0.0, sumScore = 0.0,scoreAvg =
0.0, totalScore = 0.0 ;
	// used to store 8 scores from 8 judges, reset all to 0
	// else rubbish will be stored....
	double num[8] = {0};
	int i=0, j=0, stop = 0;

	while(stop != -1)
	{
		// prompt user for inputs
		printf("Enter 8 scores out of ten points separated by a space:\n");
		// store all the input in num[]
		for(i=0;i<8;i++)
		{
			// using %f is failed, use lf instead for double
			scanf_s("%lf", &num[i]);
			// sum up all the score
			sumScore = sumScore + num[i];
		}

		// set initial value minScore to the first array element
		minScore = num[0];
		// iterate, compare for max and min score and store them
		for(j = 0;j< 8; j++)
		{
			if( minScore > num[j])
			{
				minScore = num[j];
			}
			if( maxScore < num[j])
			{
				maxScore = num[j];
			}
		}

		// discard the lowest and highest scores
		totalScore = sumScore - (maxScore + minScore);
		// find the average score, the number of scores = 8.0 – 2.0 = 6.0
		scoreAvg = totalScore / 6.0;
		// print all the related information
		printf("\n====================================\n");
		printf("Your Lowest score is %.2f\n", minScore);
		printf("Your Maximum score is %.2f\n", maxScore);
		printf("Your Total point is %.2f\n", totalScore);
		printf("Your average point is %.2f\n", scoreAvg);
		printf("====================================\n");
		printf("=========CONGRATULATION!=============\n");
		// ask for more participant....
		printf("More participant? -1 to stop, other to continue: ");
		scanf_s("%d", &stop);
	}
	return 0;
```

}

A sample output:



10. Write a program that allows the user to enter students' names followed by their test scores and outputs the following information (assume that maximum number of students is 50):

    a. The average score.
    b. Names of all students whose test scores are below the average, with an appropriate message.
    c. Highest test score and the name of all students having the highest score.

Answer: Use 2 arrays to store the student names and scores respectively and then manipulate the array contents.

```
//-----------------------------------------------------------
// Calculate student score and basic report
//-----------------------------------------------------------
#include <stdio.h>

int main()
{
	// an array of double to store student's score
	double studentscore[50];
	// a 2D array of string to store student's name
	char studentname[50][50];
	double studentavg = 0.0, sumscore = 0.0, averagescore = 0.0,
highestscore =0.0;
	// index and terminal variables
	int i = 0, stop = 0, k = 0;

	// read and store student name and score
	do
	{
		printf("Enter student name: ");
		// null terminated string, scanf_s() only accept 1 string
		// can try gets()/gets_s()
		scanf_s("%s", &studentname[i], sizeof(studentname[i]));
		printf("Enter student score: ");
```

```
      scanf_s("%lf", &studentscore[i]);
      // increment the array index
      i++;
      // continue for next data?
      printf("More data? -1 to stop, others to continue: ");
      scanf_s("%d", &stop);
}  while(stop != -1);

// some cosmetic...
printf("\n=================REPORT==================\n");
printf("Student Name\tScore\n");
printf("------------\t-----\n");
// set initial value of the highest score to the 1st array element
// and then compare 1 by 1 in the for loop...
highestscore = studentscore[0];
// the i index less 1, coz we increment after storing it
// in the do-while loop...
for(k=0;k<=i-1;k++)
{
      // print all the student names and their respective scores
      printf("%s\t\t%.2f\n",studentname[k],studentscore[k]);
      // summing up all the score for average calculation
      sumscore = sumscore + studentscore[k];
      // determining the highest score
      if(highestscore < studentscore[k])
            highestscore = studentscore[k];
}

// calculate class average score
printf("\nThe number of student is %d\n",i);
averagescore = sumscore / i;
printf("The average score for this class is %.2f\n", averagescore);
// some cosmetic formatting...
printf("\n================================================\n");
printf("Below The Average Students! Work Harder!\n");
printf("================================================\n");
printf("\nStudent Name\tScore\n");
printf("------------\t-----\n");

// list down all the below average students
for(k=0;k<=i-1;k++)
{
      if(studentscore[k] < averagescore)
            printf("%s\t\t%.2f\n", studentname[k], studentscore[k]);
}

// some cosmetic formatting...
printf("\n================================================\n");
printf("Top Scorer Student! Congratulation!\n");
printf("================================================\n");
printf("\nStudent Name\tScore\n");
printf("------------\t-----\n");

// list down all the highest mark students
for(k=0;k<=i-1;k++)
{
      if(studentscore[k] == highestscore)
            printf("%s\t\t%.2f\n", studentname[k], studentscore[k]);
```

```
        }
        return 0;
}
```

A sample output: