Lab 05

Nested Loops and Control Flow using ‘break’ and ‘continue’

Muhammad Mujtaba

CMD ID: 540040

mmujtaba.bese25seecs@seecs.edu.pk

Class: BESE 16B

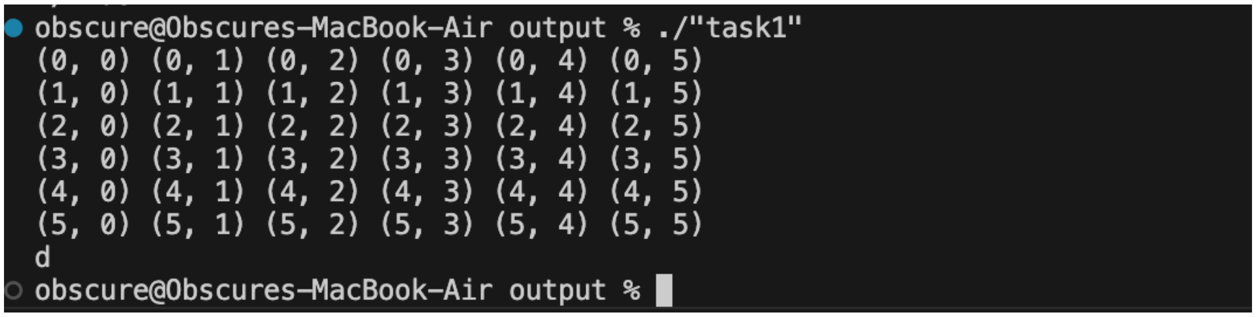
Batch: 2k25

Task 1 [CLO 1]:

CODE:

```cpp
#include <iostream>

int main()
{
    int min_x = 0;
    int max_x = 5;
    int min_y = 0;
    int max_y = 5;

    for (int i = min_x; i <= max_x; i++)
    {
        for (int j = min_y; j <= max_y; j++)
        {
            std::cout << "(" << i << ", " << j << ") ";
        }
        std::cout << "\n";
    }

    std::cin.ignore();
    std::cin.get();
    return 0;
}
```

OUTPUT:



```
obscure@Obscures-MacBook-Air output % ./"task1"
(0, 0) (0, 1) (0, 2) (0, 3) (0, 4) (0, 5)
(1, 0) (1, 1) (1, 2) (1, 3) (1, 4) (1, 5)
(2, 0) (2, 1) (2, 2) (2, 3) (2, 4) (2, 5)
(3, 0) (3, 1) (3, 2) (3, 3) (3, 4) (3, 5)
(4, 0) (4, 1) (4, 2) (4, 3) (4, 4) (4, 5)
(5, 0) (5, 1) (5, 2) (5, 3) (5, 4) (5, 5)
d
obscure@Obscures-MacBook-Air output %
```

Task 2 [CLO 1]:

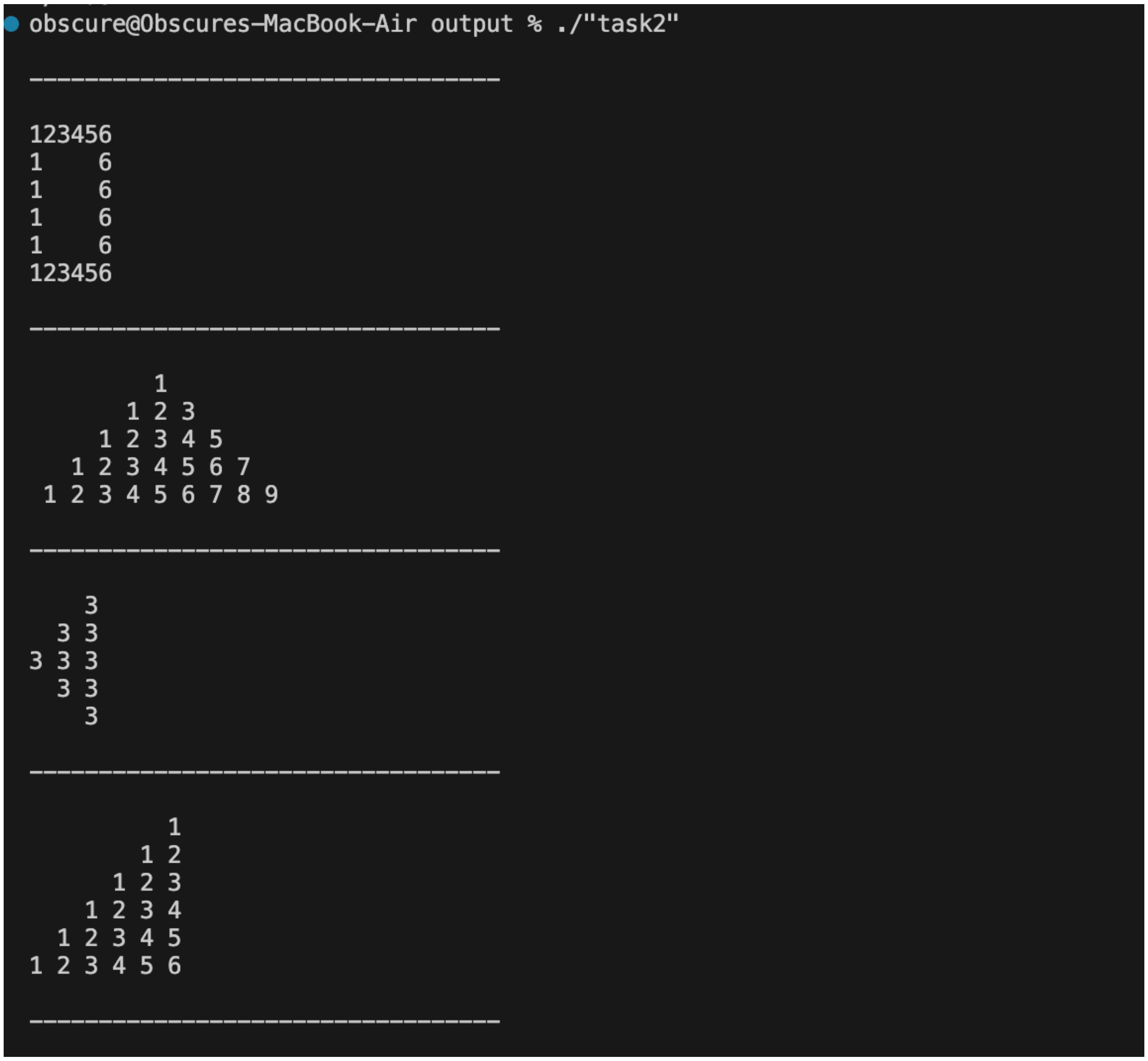
CODE:

```cpp
#include <iostream>
#include <math.h>
#include <iomanip>

// utility for printing spaces
inline void print_spaces(int count, bool debug = false)
{
    // `count + 1` is used as std::setw sets the total width including the first charcter of
next output field
    std::cout << std::setw(count + 1) << std::setfill(debug ? '.' : ' ');
}

void numberSquarePattern(int size);
void pyramidPattern(int lines_y);
void rightAlignedPyramidPattern(int lines_y, char toPrint);
void rightTrianglePattern(int size);

int main()
{
    std::cout << "\n-----------------------------------\n\n";
    numberSquarePattern(6);
    std::cout << "\n-----------------------------------\n\n";
    pyramidPattern(5);
    std::cout << "\n-----------------------------------\n\n";
    rightAlignedPyramidPattern(5, '3');
    std::cout << "\n-----------------------------------\n\n";
    rightTrianglePattern(6);
    std::cout << "\n-----------------------------------\n\n";

    // ignore previous input
    std::cin.ignore();
    std::cin.get();
    return 0;
}

void numberSquarePattern(int size)
{
    // size is the size of square

    // at every place inside the region "size x size"
    // it only prints the character if it is at outer edge

    for (int i = 1; i <= size; i++)
    {
        for (int j = 1; j <= size; j++)
        {
            // if first or last line, print all line
            if (i == 1 || i == size)
                std::cout << j;
            else
            {
                // if first or last item IN line, print "the number" otherwise print " "
                if (j == 1 || j == size)
                    std::cout << j;
                else
                    std::cout << " ";
            }
        }
        std::cout << "\n";
    }
}
```

```cpp
void pyramidPattern(int lines_y)
{
    // lines_y is how many lines to print in vertical direction
    const int height = lines_y * 2; // height is two times "lines_y" as we skip even number
lines
    // no cost as compiler will optimize this away

    // first calculate how many spaces are required for the character
    // then print the character just like a right facing triangle
    // CAN BE THOUGHT AS
    // first print right facing triangle then slant it
    // but processes on each line seperately

    int spaces_count;
    for (int i = 1; i <= height; i += 2) // `* 2` and `+= 2` as we only need to print even
numbers
    {
        spaces_count = height - i;
        print_spaces(spaces_count);
        for (int j = 1; j <= height; j++)
        {
            if (j <= i)
                std::cout << j << " ";
        }
        std::cout << "\n";
    }
}

void rightAlignedPyramidPattern(int lines_y, char toPrint)
{
    // if even number of lines are being printed we will print the middle line twice
    const int lines_y_half_ceiling = std::ceil(lines_y / 2.f); // pre calculate
    // will be optimzed away by compiler

    // same priciple as above
    // calculate spaces then print character

    int spaces_count;
    for (int i = 1; i <= lines_y; i++)
    {
        // print spaces
        spaces_count = std::abs(lines_y_half_ceiling - i) * 2;
        if (lines_y % 2 == 0 && i > lines_y_half_ceiling)
            spaces_count -= 2;
        print_spaces(spaces_count);
        // print characters
        for (int j = 1; j <= lines_y; j++)
        {
            // before middle point
            if (i <= lines_y_half_ceiling)
            {
                // print triangle facing left
                if (j <= i)
                    std::cout << toPrint << " ";
            }
            else
            {
                // print triangle facing right
                if (j >= i)
                    std::cout << toPrint << " ";
            }
        }
        std::cout << "\n";
    }
}
```

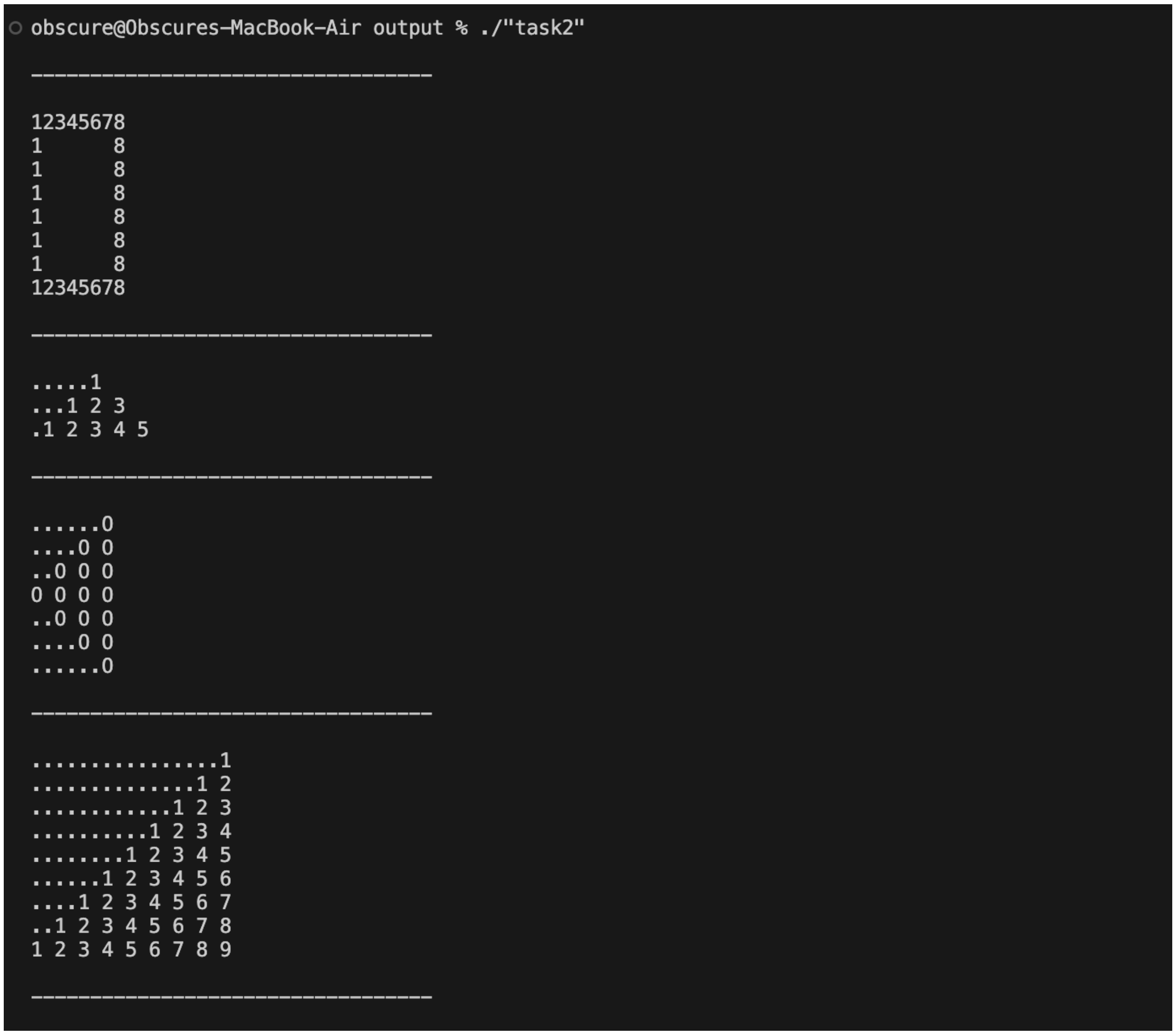
```
void rightTrianglePattern(int size)
{
    int spaces_count;
    for (int i = 1; i <= size; i++)
    {
        spaces_count = size - i;
        print_spaces(spaces_count * 2);
        for (int j = 1; j <= size; j++)
        {
            // print triangle facing left
            if (j <= i)
                std::cout << j << " ";
        }
        std::cout << "\n";
    }
}
```

OUTPUT:



```
obscure@Obscures-MacBook-Air output % ./"task2"

----------------------------------

123456
1    6
1    6
1    6
1    6
123456

----------------------------------

         1
       1 2 3
     1 2 3 4 5
   1 2 3 4 5 6 7
 1 2 3 4 5 6 7 8 9

----------------------------------

    3
  3 3
3 3 3
  3 3
    3

----------------------------------

          1
        1 2
      1 2 3
    1 2 3 4
  1 2 3 4 5
1 2 3 4 5 6

----------------------------------
```

```cpp
// settings `DEFAULT` //
inline void print_spaces(int count, bool debug = false)
numberSquarePattern(6);
pyramidPattern(5);
rightAlignedPyramidPattern(5, '3');
rightTrianglePattern(6);
```



```
obscure@Obscures-MacBook-Air output % ./"task2"

 ----------------------------------

 12345678
 1      8
 1      8
 1      8
 1      8
 1      8
 1      8
 12345678

 ----------------------------------

 .....1
 ...1 2 3
 .1 2 3 4 5

 ----------------------------------

 ......0
 ....0 0
 ..0 0 0
 0 0 0 0
 ..0 0 0
 ....0 0
 ......0

 ----------------------------------

 ................1
 ..............1 2
 ............1 2 3
 ..........1 2 3 4
 ........1 2 3 4 5
 ......1 2 3 4 5 6
 ....1 2 3 4 5 6 7
 ..1 2 3 4 5 6 7 8
 1 2 3 4 5 6 7 8 9

 ----------------------------------
```

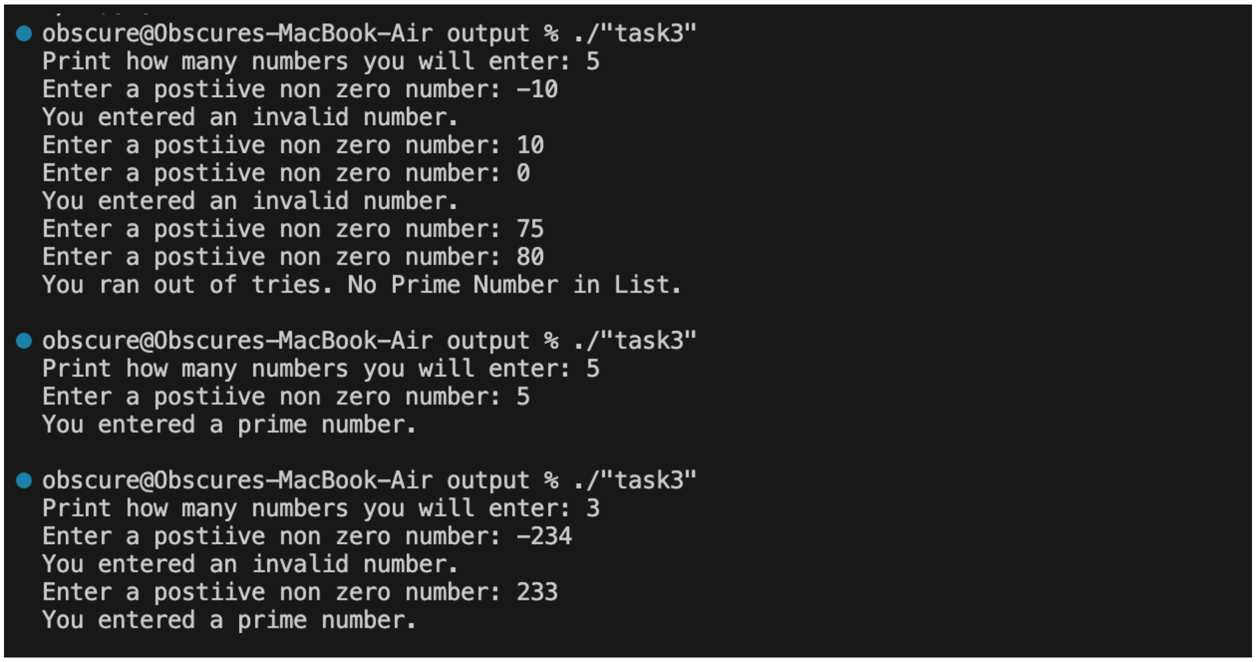
```cpp
// settings `ALTERED INPUT` //
inline void print_spaces(int count, bool debug = true)
numberSquarePattern(8);
pyramidPattern(3);
rightAlignedPyramidPattern(7, '0');
rightTrianglePattern(9);
```

Task 3 [CLO 2]:

CODE:

```
#include <iostream>

bool isPrime(int number, bool &primeError);

int main()
{
    int count;

    std::cout << "Print how many numbers you will enter: ";
    std::cin >> count;

    int number;              // the number
    bool primeFound = false; // is primeNumber Found
    bool primeError = false; // is error detected
    while (count > 0)
    {
        count--;

        std::cout << "Enter a postiive non zero number: ";
        std::cin >> number;
        if (isPrime(number, primeError))
        {
            std::cout << "You entered a prime number.\n";
            primeFound = true;
            break;
        }

        if (primeError)
            std::cout << "You entered an invalid number.\n";
    }

    if (count <= 0 && !primeFound)
        std::cout << "You ran out of tries. No Prime Number in List.\n";

    std::cin.ignore();
    std::cin.get();
    return 0;
}

bool isPrime(int number, bool &primeError)
{
    // if number is negative or zero then set primeError to true
    if (number <= 0)
    {
        primeError = true;
        return false;
    }

    if (number == 1 || number == 2)
        return true;

    primeError = false;
    // check numbers from 2 to (number / 2) + 1 for divisibility
    for (int i = 2; i < int(number / 2.f) + 1; i++)
    {
        if (number % i == 0)
            return false;
    }

    return true;
}
```

OUTPUT:



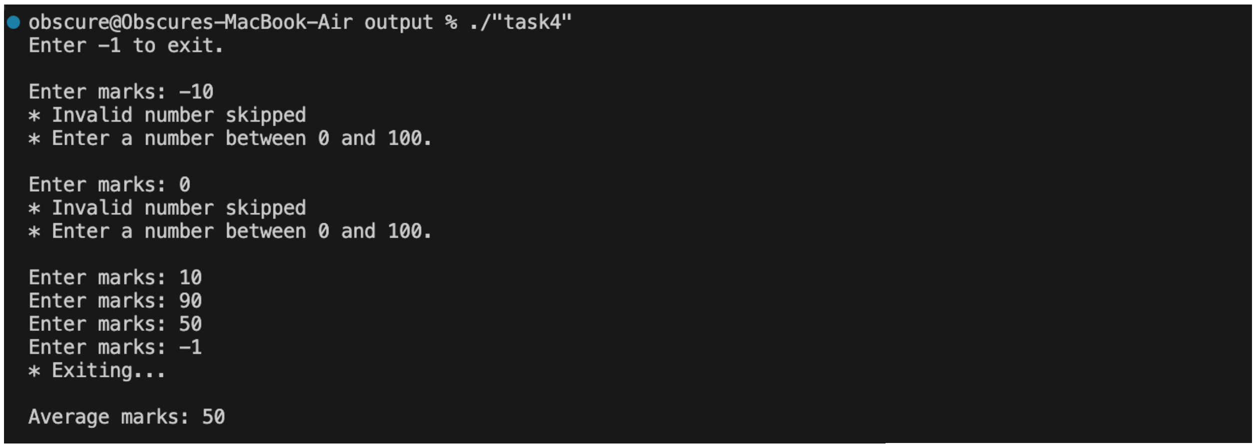
```
● obscure@Obscures-MacBook-Air output % ./"task3"
  Print how many numbers you will enter: 5
  Enter a postiive non zero number: -10
  You entered an invalid number.
  Enter a postiive non zero number: 10
  Enter a postiive non zero number: 0
  You entered an invalid number.
  Enter a postiive non zero number: 75
  Enter a postiive non zero number: 80
  You ran out of tries. No Prime Number in List.

● obscure@Obscures-MacBook-Air output % ./"task3"
  Print how many numbers you will enter: 5
  Enter a postiive non zero number: 5
  You entered a prime number.

● obscure@Obscures-MacBook-Air output % ./"task3"
  Print how many numbers you will enter: 3
  Enter a postiive non zero number: -234
  You entered an invalid number.
  Enter a postiive non zero number: 233
  You entered a prime number.
```

Task 4 [CLO 2]:

CODE:

```cpp
#include <iostream>

int main()
{
    float sum = 0;
    int count = 0;

    int tmp_input;
    std::cout << "Enter -1 to exit. \n\n";
    while (true)
    {
        std::cout << "Enter marks: ";
        std::cin >> tmp_input;

        // if input -1 then exit
        if (tmp_input == -1)
        {
            std::cout << "* Exiting... \n\n";
            break;
        }

        // check if between 0 and 10
        if (tmp_input <= 0 || tmp_input >= 100)
        {
            std::cout << "* Invalid number skipped \n";
            std::cout << "* Enter a number between 0 and 100. \n\n";
            continue; // take another numb from
        }

        sum += tmp_input;
        count++;
    }

    std::cout << "Average marks: " << sum / count << "\n";

    std::cin.ignore();
    std::cin.get();
    return 0;
}
```

OUTPUT:



```
obscure@Obscures-MacBook-Air output % ./"task4"
Enter -1 to exit.

Enter marks: -10
* Invalid number skipped
* Enter a number between 0 and 100.

Enter marks: 0
* Invalid number skipped
* Enter a number between 0 and 100.

Enter marks: 10
Enter marks: 90
Enter marks: 50
Enter marks: -1
* Exiting...

Average marks: 50
```