higher education & training

Department:
Higher Education and Training
REPUBLIC OF SOUTH AFRICA

MARKING GUIDELINE

NATIONAL CERTIFICATE (VOCATIONAL)

INTRODUCTION TO TECHNICAL PROGRAMMING
NQF LEVEL 3
(Second paper)

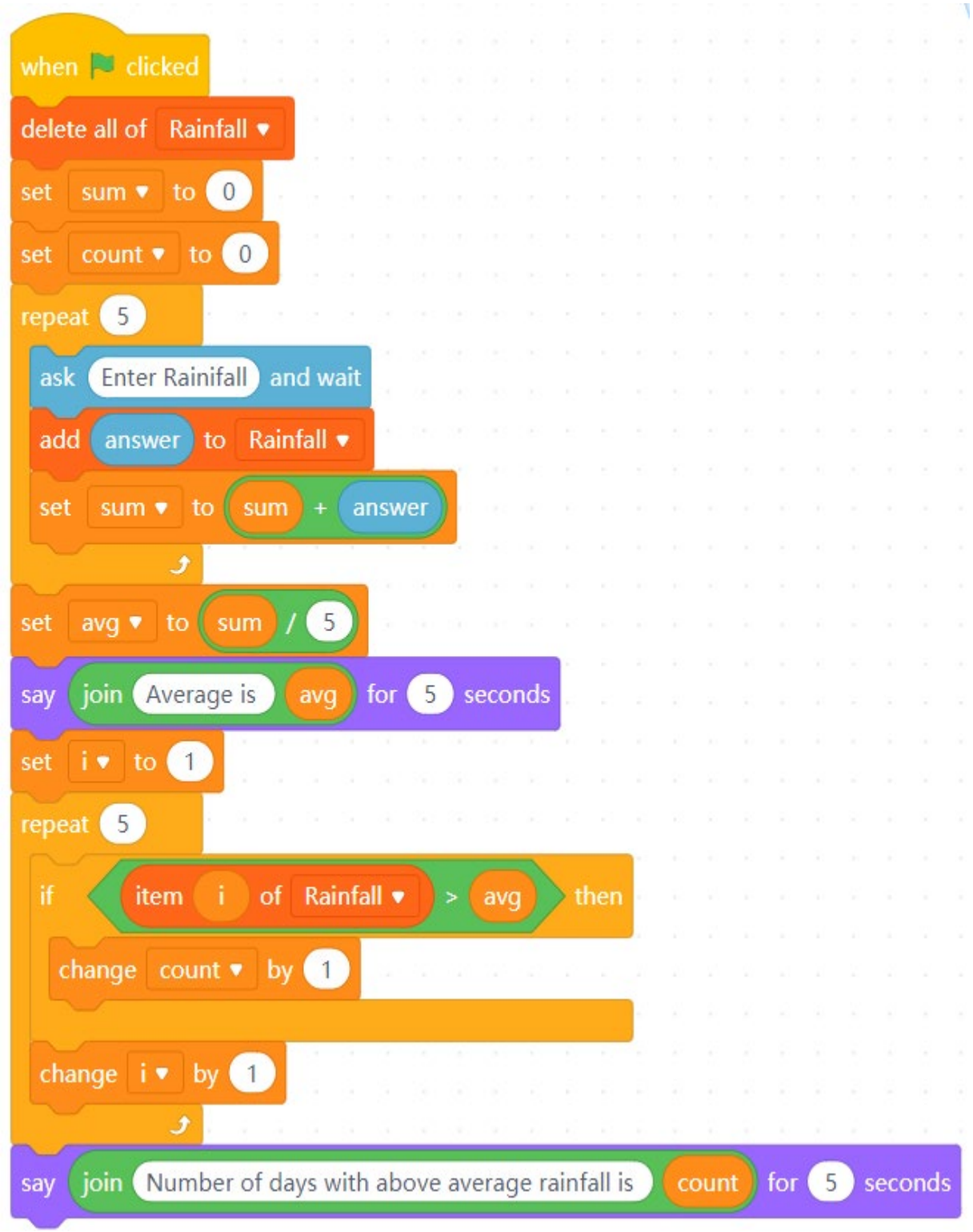
5 December 2024

This marking guideline consists of 8 pages.

Copyright reserved

QUESTION 1

Sample solution



Mark allocation

1.1 1 Mark for creating application. (1)

1.2 1 Mark for creating list called Rainfall. (1)

1.3 2 Marks for declaring ALL variables. (2)

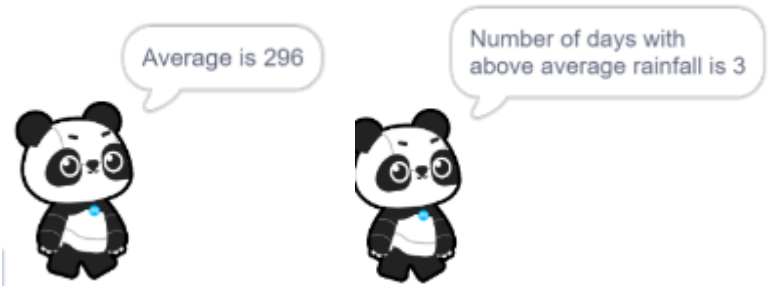
1.4 1 Mark for using a loop that runs **5** times.
1 Mark for input. (2)

1.5 1 Mark for calculating sum
1 Mark for calculating average
1 Mark for displaying average. (3)

1.6 1 Mark for loop
1 Mark for IF condition
1 Mark for counting days above average (3)

1.7 1 Mark for printout of code. (1)

1.8 1 Mark for printout of correct average.
1 Mark for printout of correct number of days.



(2)
[15]

QUESTION 2

The student can draw the shape using nested or single loop. A guideline of the solution of both methods are provided below.

2.1	// Examination Number	✓	(1)
2.2	int rows; cout << "Please enter the number of rows " << endl; cin >> rows;	✓ ✓ ✓	(3)

2.3	for (int i = rows ✓ ; i >= 1 ✓ ; i -- ✓)
	{
	for (int j = i; ✓ j>= 1 ✓; j-- ✓)
	{
	cout << j << "#"; ✓✓
	}
	cout << endl; ✓
	}
			(9)

[13]

QUESTION 3

Refer to the sample solution below on how to allocate the marks. The students attempt can differ from the sample solution provided. Marks are to be allocated as per the instructions on the question paper.

3.1	#include <cstring> or #include <string>		✓	(1)
3.2	void reverseIT();		✓	(1)

3.3	string sOriginal = ""; // Initialize to an empty string
	string sReverse = "";
	
	1 mark for the declaration and 1 mark for the initialisation
			✓✓	(2)

3.4	In the main function:			
	3.4.1	cout << "Message encryption program\n\n";	✓	(1)
	3.4.2	getline(cin, sOriginal);	✓✓	(2)
	3.4.3	reverseIT();	✓	(1)
	3.4.4	cout << "Original message: " << sOriginal << endl; cout << "Reversed (encrypted) message: " << sReverse << endl;	✓ ✓	(2)

3.5	void reverseIT()
	{
	// Reverse the string using a loop
	for (int i = sOriginal.length() - 1; i >= 0; i--) {
	sReverse = sReverse + sOriginal[i];
	}
	}
	
	Mark allocation:
	• 1 Mark for the function header
	• 2 Marks for traversing backwards
	• 1 Mark for using the length() method
	• 1 Mark for using the [] operator to access a single character
	• 2 mark for building up concatenating the reverse string
				(7)

[17]

QUESTION 4

Refer to the sample solution below on how to allocate the marks. The students attempt can differ from the sample solution provided. Marks are to be allocated as per the instructions on the question paper.

4.1

```
void displayPriceList() {
    cout << "Product Pricing Information:" << endl;
    cout << "Code | Product       | Unit Price (Rands) | >100   | >500" << endl;
    cout << "-----|---------------|--------------------|--------|------" << endl;
    cout << "A    | LEDs 10 Pack  | 2.85               | 2.28   | 1.9" << endl;
    cout << "B    | Resistors x10 | 0.95               | 0.76   | 0.54" << endl;
}
```

✓ ✓✓ (3)

4.2

```
// Global variables
char productCode = ' ';
int quantity = 0;
string product = "";
```

✓ (1)

4.3

```
double calculatePrice() {
   double unitPrice;

   if ((productCode == 'A') || (productCode == 'a')) {
      product = "LEDs 10 Pack";
      if (quantity > 500) {
         unitPrice = 1.9;
      } else if (quantity > 100) {
         unitPrice = 2.28;
      } else {
         unitPrice = 2.85;
      }
   } else if ((productCode == 'B') || (productCode == 'b')) {
      product = "Resistors x10";
      if (quantity > 500) {
         unitPrice = 0.54;
      } else if (quantity > 100) {
         unitPrice = 0.76;
      } else {
         unitPrice = 0.95;
      }
   }

   return unitPrice * quantity;
}
```

Mark allocation:

- 1 Mark for the function header with return type
- 1 Mark for using a local variable
- 2 Marks for allocating and or assigning the name of the product to a variable – ** This can also be done in the main program (for which the marks should also be allocated)
- 2 Marks for the if (in relation to the product code A or B)

	• 3 Marks for the correct nested if or structure to determine the correct unit price based on the quantity (A) • 2 Marks for the correct nested if or structure to determine the correct unit price based on the quantity (B) • 2 Mark for returning unitPrice * quantity;			(13)
4.4	In the main function:			
	4.4.1	displayPriceList();	✓	(1)

	4.4.2	cout << "\nEnter the product code (A, B): ";
		cin >> productCode;
		productCode = toupper(productCode);
		
		cout << "Enter the quantity: ";
		cin >> quantity;
		
		while (quantity <= 0 || quantity >= 1000) {
		cout << "Invalid quantity." << endl;
		cout << "Enter the quantity: ";
		cin >> quantity;
		}
		
		Mark allocation:
		• 1 Mark display and enter product code
		• 2 Marks for catering for a or A, or b, or B (mark can also be awarded if done as part of the calculatePrice() function
		• 1 Mark display and enter quantity
		• 4 Marks for loop with condition – message and re-entering of marks
				(8)

	4.4.3	double totalPrice = calculatePrice(); cout << "Total Price for " << quantity << " " << product << " is R" << fixed << setprecision(2) << totalPrice << endl; **Mark allocation:** • 2 Marks for the function call • 2 marks for displaying the quantity and the name of the product • 1 Mark for including <iomanip> • 1 Mark for fixed point format • 1 Mark for setprecision(2)		(7)

[33]

QUESTION 5

Refer to the sample solution below on how to allocate the marks. The students attempt can differ from the sample solution provided. Marks are to be allocated as per the instructions on the question paper.

5.1	#include <cstring> or #include <string> #include <cstdlib> #include <ctime> //Declared globally char consonants[21] = {'B', 'C', 'D', 'F', 'G', 'H', 'J', 'K', 'L', 'M', 'N', 'P', 'Q', 'R', 'S', 'T', 'V', 'W', 'X', 'Y', 'Z'};	✓ ✓ ✓ ✓ ✓	(5)

5.2	// Seed the random number generator
	srand(time(0));
	
	cout << "10 Generated Number Plates:" << endl << endl;
	
	for (int i = 0; i < 10; i++) {
	string plate;
	
	// Generate first two consonants
	plate += consonants[rand() % 21];
	plate += consonants[rand() % 21];
	plate += " ";
	
	// Generate two-digit number
	int number = rand() % 100;
	
	// Append 0 if < 10
	if (number < 10) {
	plate += "0";
	}
	plate += to_string(number);
	
	plate += " ";
	// Generate last two consonants
	plate += consonants[rand() % 21];
	plate += consonants[rand() % 21];
	
	// Add GP
	plate += " GP";
	
	cout << plate << endl;
	}
			(17)

	Mark allocation: • 1 Marks for seeding the random number generator • 1 Mark for the heading • 2 Marks for the loop construct to generate 10 plates • 3 Marks for generating the first two characters randomly from the consonant list • 1 Mark for concatenating a space • 2 Marks for generating a value 0 to 99 • 2 Marks for checking <10 and to append 0 if required • 2 Marks for using the to_string function to append (concatenate the numerical values to the string) • 1 Mark for generating the next two characters randomly from the consonant list • 1 Mark for concatenating GP • 1 Mark for the display		

[22]

TOTAL: 100

Copyright reserved