Άσκηση 1

Το αρχείο `1.cpp` αφορά την προσέγγιση της προγραμματιστικής λύσης πάνω στο Θέμα 1, και αρχείο `1.exe` είναι η εκτελέσιμη λύση, στην οποία βρίσκεται η υλοποίηση της προσέγγισης. Τόσο το προγραμματιστικό, όσο και το εκτελέσιμο κομμάτι της λύσης, υλοποιήθηκε με τα εργαλεία της γλώσσας C++. Η εκτέλεση του προγράμματος πραγματοποιήθηκε με το Command Prompt & το Windows PowerShell.

1. ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΟ ΚΟΜΜΑΤΙ:

Στις πρώτες τέσσερις γραμμές του κώδικα αναγράφεται η συμπερίληψη των βοηθητικών βιβλιοθηκών της C++, όπως η `<regex>`, η `<string>`, αλλά και η `<stack>`.

Ύστερα, καθώς δεν υπάρχει κάποια **έτοιμη συνάρτηση για την προβολή της στοίβας, δημιουργήθηκε** μία **από εμάς**, η οποία **προβάλλει, σε μία γραμμή, το περιεχόμενο της στοίβας**. Στην **Εικόνα 1.1**, βλέπουμε πως η συνάρτηση παίρνει ως όρισμα μια στοίβα χαρακτήρων και προβάλλει, από την κορυφή της, ένα-ένα τα στοιχεία. Κάθε φορά, που προβάλλουμε ένα στοιχείο, το βγάζουμε από την στοίβα (του ορίσματος κι όχι από την κανονική), ώστε να προβάλλουμε το επόμενό του. Η συνάρτηση **προβάλλει τα στοιχεία από πάνω προς τα κάτω**, δηλαδή **το στοιχείο στην κορυφή, θα προβληθεί πρώτο**.

Εικόνα 1.1: Ο κώδικας της συνάρτησης DisplayStack.

```
8   //need this function to display the stack below.
9   void DisplayStack(stack<char> s)
10  {
11      if (s.empty())
12          cout << "(empty)";
13
14      //display the stack with the current elements available in one line.
15      while (!s.empty())
16      {
17          cout << s.top();
18          s.pop();
19      }
20  }
```

Καθώς προχωράει παρακάτω το πρόγραμμα, οι γραμμές 25-28 αφορούν την προβολή των παραδοχών στην υλοποίηση της άσκησης. **Οι παραδοχές έχουν ως εξής:**

- Θεωρούμε q_1 μια μη τερματική κατάσταση του ΝΑΣ.
- Θεωρούμε q_2 μια τερματική κατάσταση του ΝΑΣ.
- Στην αρχή του προγράμματος, **εισάγεται ένα στοιχείο 'n' στη στοίβα. Αποδεκτή έκφραση θα υπάρχει, αν και μόνο αν η στοίβα έχει ως μοναδικό στοιχείο το 'n'.** Στην περίπτωση, που η στοίβα βρίσκεται σε οποιαδήποτε άλλη κατάσταση, η έκφραση απορρίπτεται.

Όλη η συνάρτηση της main **τρέχει σε μία τεράστια επανάληψη τύπου do-while**. Στην αρχή της επανάληψης, προβάλλεται το μήνυμα, στο οποίο ο χρήστης θα δώσει την έκφρασή του. Για να μην υπάρξουν προβλήματα κατά την ανάγνωση της δοθείσης έκφρασης, το πρόγραμμα μετατρέπει, αυτόματα, την έκφραση σε πεζά γράμματα.

Εικόνα 1.2: Οι γραμμές, που ευθύνονται για την προβολή μηνύματος εισόδου, την ίδια την είσοδο και την μετατροπή της σε πεζά γράμματα.

```
32      //this do-while loop is going to run forever, unless the user types 'stop'.
33      do
34      {
35          //get the input from the user.
36          cout << "Input: ";
37          cin >> input;
38
39          //convert the input to lowercase.
40          transform(input.begin(), input.end(), input.begin(), [](unsigned char c) { return tolower(c); });
41
```

Αφού ο χρήστης δώσει την έκφρασή του, το πρόγραμμα ελέγχει τα, κατ' ελάχιστον, **τρία πιθανά σενάρια**:

- Με τη χρήση της κανονικής έκφρασης "[xy]+", ελέχουμε αν υπάρχουν **αποκλειστικά και τουλάχιστον μία φορά, χαρακτήρες 'x' ή 'y'.**
- Αν δοθεί ακριβώς η έκφραση "stop", το πρόγραμμα **διασφαλίζει την περίπτωση του τερματισμού του με κωδικό εξόδου 0.** Επειδή, κάθε είσοδος μετατρέπεται σε πεζά γράμματα, εκφράσεις όπως "STOP" ή "StOp" είναι αποδεκτές.

- Σε οποιαδήποτε άλλη έκφραση, το πρόγραμμα βγάζει μήνυμα σφάλματος στον χρήστη **και επαναλαμβάνει τη διαδικασία του ελέγχου (νέας) εισόδου**.

ΚΥΡΙΑ ΠΕΡΙΠΤΩΣΗ

Στο πρώτο πιθανό σενάριο, δηλαδή στην περίπτωση που υπάρχουν αποκλειστικά χαρακτήρες 'x' ή 'y', αρχικοποιούνται οι βασικές μεταβλητές, με τις οποίες δουλεύει το πρόγραμμα.

Πρώτη, από όλες τις άλλες μεταβλητές, ορίζεται η στοίβα και ο αριθμός των χαρακτήρων της έκφρασης του χρήστη. Ύστερα, ορίζονται οι βοηθητικές μεταβλητές, όπως αυτή της παρούσας κατάστασης και του μηνύματος σφάλματος. Αφού γίνουν αυτά, βάζουμε και το βοηθητικό στοιχείο 'n' στη στοίβα.

Εικόνα 1.3: Αρχικοποίηση μεταβλητών.

```
42      //if the message matches the wanted input (which has to be only x and y chars), we proceed to do what the exercise defines.
43      if (regex_match(input, regex("[xy]+")))
44      {
45          //define the stack to be used, and the input length of the input (to use it in the for loop).
46          stack<char> NAS_stack;
47          int input_length = input.length();
48
49          //push a dummy element into the stack. This will help us make various tests easier. Also, keep track of the accepted input state.
50          NAS_stack.push('n');
51          string error_output = "";
52          string state = "q2";
53
```

Ύστερα, μπαίνουμε στην **επανάληψη τύπου for.** Η επανάληψη **θα εκτελεστεί όσες φορές, όσα και τα γράμματα της εισόδου.** Δηλαδή, αν ο χρήστης έχει δώσει μία έκφραση δέκα γραμμάτων, η επανάληψη εκτελείται δέκα φορές. Εντός της επανάληψης, εφ' όσον κοιτάμε κάθε γράμμα της έκφρασης, υπάρχουν δύο περιπτώσεις. Το παρόν γράμμα **θα είναι είτε χαρακτήρας 'x', είτε χαρακτήρας 'y'.**

Στην περίπτωση που είναι χαρακτήρας 'x', το βάζουμε στη στοίβα. Αν είναι χαρακτήρας 'y', βγάζουμε ένα στοιχείο (όποιο κι αν είναι αυτό) από τη στοίβα.

Γίνονται, παράλληλα, οι κατάλληλοι έλεγχοι, **ώστε η παρούσα κατάσταση, να αντιστοιχεί στην απόρριψη/έγκριση της έκφρασης.** Παραδείγματος χάριν, αν μπει ο χαρακτήρας 'x' στην στοίβα, τότε θα αλλάξει η κατάσταση σε q_1, διότι η στοίβα δεν έχει ως μοναδικό στοιχείο το 'n'.

Αφού γίνουν οι έλεγχοι, το πρόγραμμα, **εκτυπώνει τον αριθμό της επανάληψης, την παρούσα κατάσταση, το περιεχόμενο της στοίβας και το υπόλοιπο της εισόδου,** ακριβώς πριν γίνει η επόμενη επανάληψη.

Εικόνα 1.4: Οι γραμμές, που ευθύνονται για την δομή επανάληψης τύπου for.

```
54          //for every character in the input
55          for (int i = 0; i < input_length; i++)
56          {
57              //save the previous element and the current element to these variables.
58              char current_element = input.at(i);
59
60              //if the current character is an 'x', it means that it can be pushed.
61              if (current_element == 'x')
62              {
63                  NAS_stack.push(current_element);
64                  state = "q1";
65              }
66              //otherwise, if it is a 'y', it means that we have to pop one character out from the stack.
67              else if (!NAS_stack.empty())
68              {
69                  NAS_stack.pop();
70
71                  if (!NAS_stack.empty() && NAS_stack.top() == 'n')
72                      state = "q2";
73                  else
74                      state = "q1";
75              }
76
77              //after every loop, we create a substring of the input, representing the remainder of the elements to be tested.
78              //we also use DisplayStack(), which outputs the stack after every loop.
79              string remaining_input = input.substr(i + 1, input_length - 1);
80              cout << (i + 1) << ". Current State: " << state << ", Stack Items: ";
81              DisplayStack(NAS_stack);
82              cout << ", Remaining Input: " << remaining_input << endl;
83          }
```

Όταν τελειώσει η δομή επανάληψης, ελέγχουμε το κορυφαίο στοιχείο της στοίβας. Αν η στοίβα είναι άδεια ή αν η στοίβα έχει ως κορυφαίο στοιχείο έναν χαρακτήρα ‘x’, τότε απορρίπτουμε την έκφραση. Ειδάλλως, η έκφραση είναι αποδεκτή. Ύστερα εμφανίζουμε το κατάλληλο μήνυμα στον χρήστη.

Εικόνα 1.5: Η διαδικασία έγκρισης/απόρριψης της έκφρασης.

```
85          if (NAS_stack.empty())
86          {
87              error_output = "Stack is empty.";
88              state = "q1";
89          }
90          else if (NAS_stack.top() == 'x')
91          {
92              error_output = "Element 'n' is either absent or not the top element.";
93              state = "q1";
94          }
95
96          //here is the final check. If the stack's top element is our dummy character, it means that the input is correct
97          if (state.compare("q2") == 0)
98              cout << endl << "String '" << input << "' is accepted." << endl << endl;
99          //for any other state of the stack, the input is incorrect. If the top element happens to be something else than 'n', the input is false.
100         else
101             cout << endl << "String '" << input << "' is NOT accepted." << endl << "Error message: " << error_output << endl << endl;
102     }
```

ΟΡΘΟΣ ΤΕΡΜΑΤΙΣΜΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

Αν ο χρήστης γράψει με οποιαδήποτε μορφή τη λέξη “stop”, **το πρόγραμμα τερματίζεται με κωδικό εξόδου 0.** Αυτό συμβαίνει, διότι είμαστε σε μία δομή επανάληψης τύπου **do-while** με συνθήκη **true**. Πράγμα, που σημαίνει ότι το πρόγραμμα, υπό κανονικές συνθήκες, **δεν θα σταματήσει ποτέ να τρέχει.** Για αυτόν τον λόγο, όποτε ο χρήστης γράψει τη λέξη ‘stop’, **«σπάμε» την επανάληψη με την εντολή break**. Ύστερα, ακολουθεί ένα μήνυμα, το οποίο ενημερώνει τον χρήστη για την ορθή λήξη του προγράμματος.

Εικόνα 1.6: Ορθός τερματισμός του προγράμματος με τη χρήση της εντολής break.

```
104         //if the user wants to stop the program, break out of the loop, then exit.
105         else if (input.compare("stop") == 0)
106         {
107             break;
108         }
```

```
118     //at this point, there's nothing else to be done. exit with code 0.
119     cout << "Program stopped regularly." << endl;
120     return 0;
```

ΜΗ ΟΡΘΗ ΕΙΣΟΔΟΣ

Αν, η είσοδος του χρήστη, δεν υποπέσει σε κάποια από τις παραπάνω περιπτώσεις, τότε πρόκειται για μια εσφαλμένη έκφραση. Σε αυτήν την περίπτωση, το πρόγραμμα εκτυπώνει το ανάλογο μήνυμα στον χρήστη και επαναλαμβάνει την δομή της do-while.

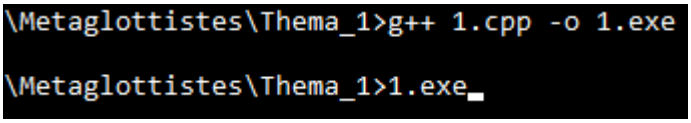
Εικόνα 1.7: Περίπτωση μη ορθής εισόδου.

```
110         //if the input doesn't match any case above, there must be some mistake. Inform the user accoordingly.
111         else
112         {
113             cout << "Incorrect Input. Only x or y characters are allowed. If you want to exit, type 'stop'." << endl << endl;
114         }
115
116     } while (true);
```

2. ΕΚΤΕΛΕΣΙΜΟ ΚΟΜΜΑΤΙ

Με τη χρήση του μεταγλωττιστή του GNU, στο λειτουργικό σύστημα των Windows, πραγματοποιήθηκε η μεταγλώττιση του αρχείου 1.cpp. Για να γίνει αυτό, χρειάστηκε, πρώτα, μέσω ενός τερματικού, να γραφεί ακριβώς η εντολή `g++ 1.cpp -o 1.exe`, ώστε να παράγουμε το εκτελέσιμο αρχείο.

Εικόνα 2.1: Η εντολή που χρειάζεται, για να παραχθεί το εκτελέσιμο αρχείο, εκτέλεση της οποίας δεν θα έπρεπε να επιφέρει κάποιο σφάλμα ή προειδοποίηση.

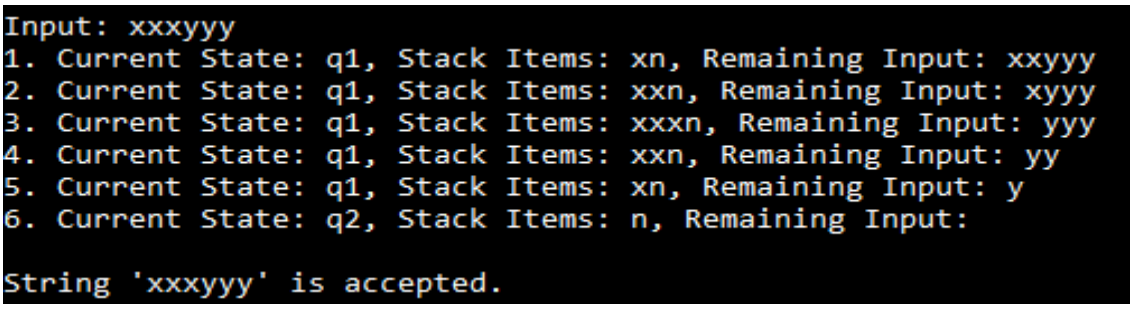


Το αρχείο 1.exe δεν είναι απαραίτητο να ανοιχτεί μέσω κάποιου τερματικού, διότι σε αυτό το σημείο, φαίνεται, εμπράκτως, η χρήση της «ατέρμονης» επανάληψης do-while. Εφ' όσον το πρόγραμμα δεν θα τερματιστεί ποτέ, χωρίς της βούληση του χρήστη, το εκτελέσιμο αρχείο μπορεί να ανοιχθεί, ακόμη και με το πάτημα διπλού κλικ από το ποντίκι.

ΑΝΑΛΥΣΗ ΕΙΣΟΔΟΥ

Κάθε φορά που το πρόγραμμα δέχεται μια είσοδο (αποτελούμενη από χαρακτήρες 'x' και 'y'), αναλύει τα βήματα που οδήγησαν στην απόρριψη ή την αποδοχή της έκφρασης που δίνει ο χρήστης. **Δίνονται, παρακάτω, τρία παραδείγματα <u>σωστών</u> εκφράσεων.**

Εικόνα 2.2: Παράδειγμα #1 Σωστής Εκτέλεσης.



Εικόνα 2.3: Παράδειγμα #2 Σωστής Εκτέλεσης.

```
Input: xxyyxxxyyyxyxy
1. Current State: q1, Stack Items: xn, Remaining Input: xyyxxxyyyxyxy
2. Current State: q1, Stack Items: xxn, Remaining Input: yyxxxyyyxyxy
3. Current State: q1, Stack Items: xn, Remaining Input: yxxxyyyxyxy
4. Current State: q2, Stack Items: n, Remaining Input: xxxyyyxyxy
5. Current State: q1, Stack Items: xn, Remaining Input: xxyyyxyxy
6. Current State: q1, Stack Items: xxn, Remaining Input: xyyyxyxy
7. Current State: q1, Stack Items: xxxn, Remaining Input: yyyxyxy
8. Current State: q1, Stack Items: xxn, Remaining Input: yyxyxy
9. Current State: q1, Stack Items: xn, Remaining Input: yxyxy
10. Current State: q2, Stack Items: n, Remaining Input: xyxy
11. Current State: q1, Stack Items: xn, Remaining Input: yxy
12. Current State: q2, Stack Items: n, Remaining Input: xy
13. Current State: q1, Stack Items: xn, Remaining Input: y
14. Current State: q2, Stack Items: n, Remaining Input:

String 'xxyyxxxyyyxyxy' is accepted.
```

Εικόνα 2.4: Παράδειγμα #3 Σωστής Εκτέλεσης.

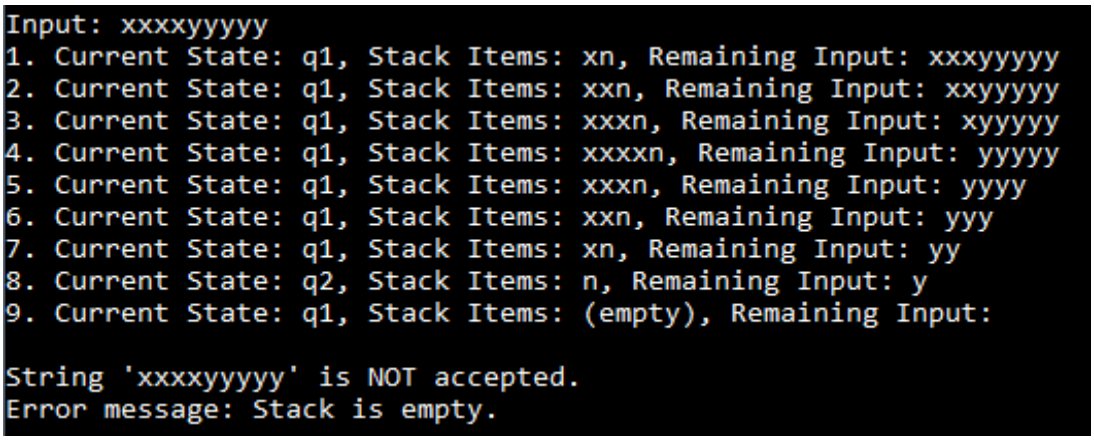
```
Input: xxxxyyxxyyyy
1. Current State: q1, Stack Items: xn, Remaining Input: xxxyyxxyyyy
2. Current State: q1, Stack Items: xxn, Remaining Input: xxyyxxyyyy
3. Current State: q1, Stack Items: xxxn, Remaining Input: xyyxxyyyy
4. Current State: q1, Stack Items: xxxxn, Remaining Input: yyxxyyyy
5. Current State: q1, Stack Items: xxxn, Remaining Input: yxxyyyy
6. Current State: q1, Stack Items: xxn, Remaining Input: xxyyyy
7. Current State: q1, Stack Items: xxxn, Remaining Input: xyyyy
8. Current State: q1, Stack Items: xxxxn, Remaining Input: yyyy
9. Current State: q1, Stack Items: xxxn, Remaining Input: yyy
10. Current State: q1, Stack Items: xxn, Remaining Input: yy
11. Current State: q1, Stack Items: xn, Remaining Input: y
12. Current State: q2, Stack Items: n, Remaining Input:

String 'xxxxyyxxyyyy' is accepted.
```

Σε όλες τις παραπάνω περιπτώσεις, **το πρόγραμμα βρέθηκε στην κατάσταση q_2, χωρίς να εκκρεμεί υπόλοιπο εισόδου, ούτε έχοντας παραπάνω ή λιγότερα στοιχεία η στοίβα.**

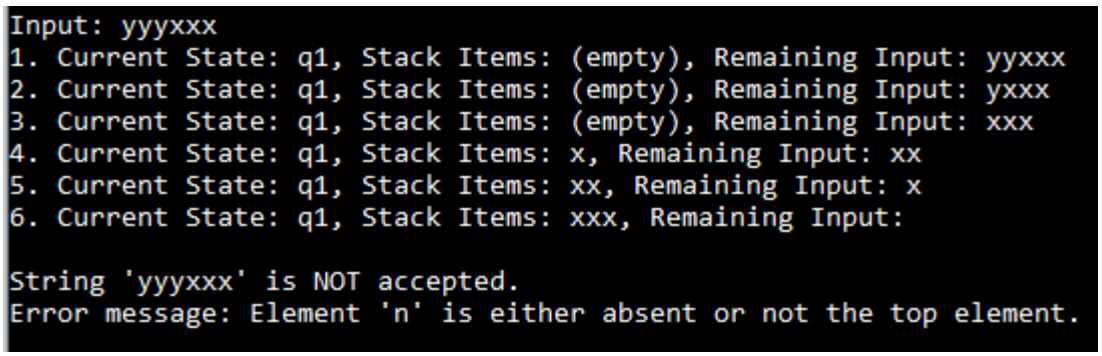
Παρατηρείται, επίσης, στο **παράδειγμα 2, ότι το αυτόματο βρέθηκε σε κατάσταση q_2, σε τέσσερις διαφορετικές περιπτώσεις** (στους αριθμούς επανάληψης 4, 10, 12 και 14). Ωστόσο, δεν είχε σημασία, διότι **εκκρεμούσε το υπόλοιπο της εισόδου.**

Αξίζει, ωστόσο, να δούμε τις περιπτώσεις, όταν κάτι πάει στραβά. **Δίνονται, παρακάτω, τρεις υποδείξεις μη ορθών εισόδων:**

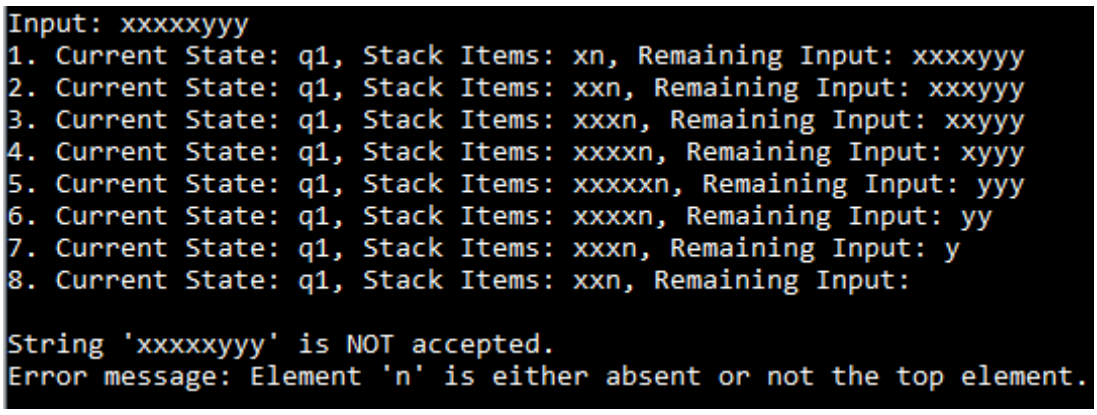
Εικόνα 2.5: Παράδειγμα #1 Εσφαλμένης Εισόδου.



Εικόνα 2.6: Παράδειγμα #2 Εσφαλμένης Εισόδου.



Εικόνα 2.7: Παράδειγμα #3 Εσφαλμένης Εισόδου.



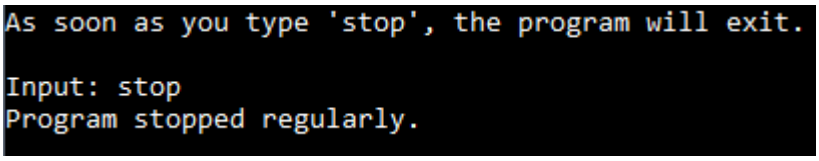
Στα δύο πρώτα παραδείγματα, **το στοιχείο 'n' απουσιάζει από τη λίστα, ενώ στο τρίτο παράδειγμα, δεν είναι το μόνο στοιχείο της στοίβας.** Ως εκ τούτου, οι δύο αυτές περιπτώσεις απορρίπτονται.

Στο **παράδειγμα 2**, **το πρόβλημα προκύπτει από την αρχή, διότι φεύγει, άμεσα, ο χαρακτήρας 'n' από τη στοίβα**. Όμως, **το πρόγραμμα δεν «κόβει» την εκτέλεσή του και συνεχίζει έως το τέλος,** επειδή ο έλεγχος της κατάστασης γίνεται στο τέλος.

ΤΕΡΜΑΤΙΣΜΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

Όπως προαναφέρθηκε και παραπάνω, **ο βέλτιστος τρόπος, που τερματίζεται το πρόγραμμα, είναι με τη χρήση της έκφρασης "stop".** Αν η εκτέλεση του προγράμματος γίνεται μέσω ενός τερματικού, ο χρήστης μπορεί να δει και το μήνυμα ορθού τερματισμού.

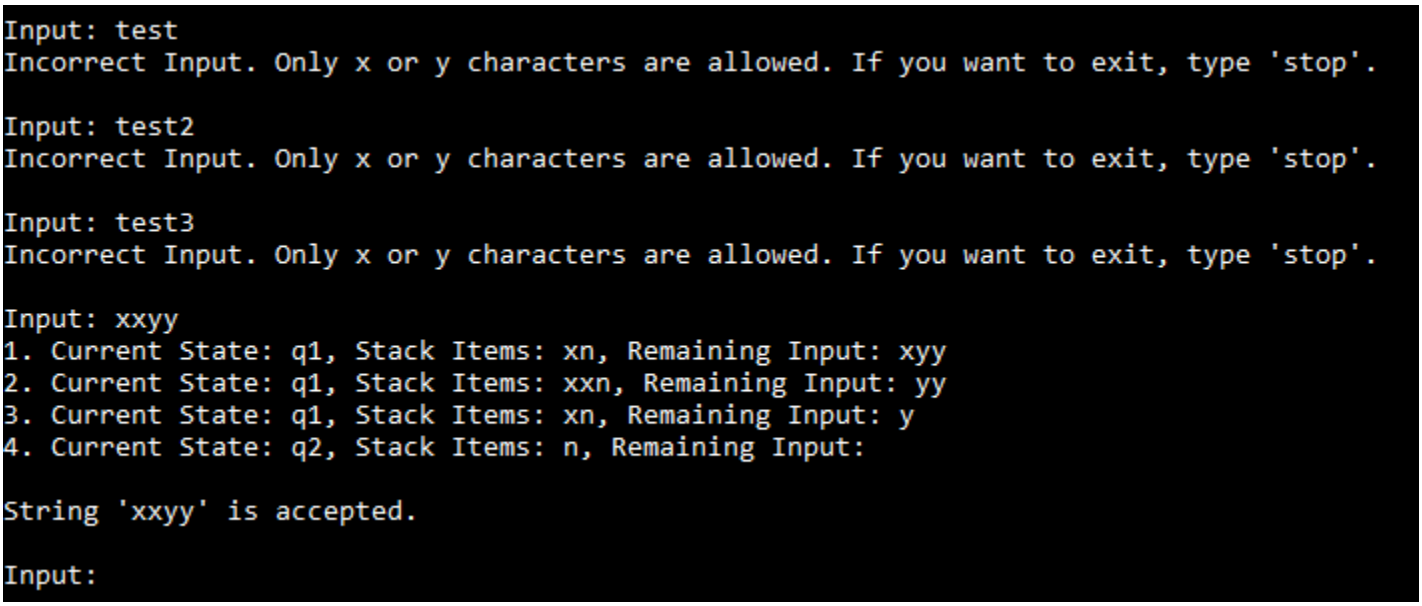
Εικόνα 2.8: Ορθός τερματισμός προγράμματος.



ΜΗ ΑΝΑΓΝΩΡΙΣΙΜΕΣ ΕΙΣΟΔΟΙ

Οι είσοδοι δεν καταγράφονται αν δεν αποτελούνται μονάχα από χαρακτήρες 'x' και 'y', πράγμα που σημαίνει ότι ο χρήστης μπορεί να βάλει όσες εισόδους επιθυμεί, μέχρι να βάλει μια σωστή. **Μια τέτοια περίπτωση (εσφαλμένης) εισόδου δεν θα βάλει το πρόγραμμα στη διαδικασία αναγνώρισης χαρακτήρων.** Οπότε, το πρόγραμμα προβάλλει ανάλογο μήνυμα στον χρήστη και ξεκινάει από την αρχή.

Εικόνα 2.9: Το πρόγραμμα δέχτηκε 3 μη αναγνωρίσιμες εισόδους, πριν προχωρήσει στην ανάλυση της σωστής εισόδου.



ΤΕΛΟΣ ΠΑΡΟΥΣΙΑΣΗΣ ΑΣΚΗΣΗΣ 1.

Συντάκτης παρουσίασης: **Γεώργιος Σεϊμένης Π19204**