Question 1

Question:
Write a program in C++ to find all the tokens in a given string separated by space. Also write the functions to classify a given token into operator or operands. The given string may contain integer or alphabets as operands.

Code:

```cpp
#include <iostream>
#include <string>
using namespace std;

// this comment is here to demonstrate an extremely long line length, well beyond what you should probably allow in your own code, though sometimes you'll be highlighting code you can't refactor, which is unfortunate but should be handled gracefully

pair < string *, int > tokenize(string str) {
  int n = str.length();
  string * tokens = new string[n];
  int i = 0;
  int j = 0;
  while (i < str.length()) {
    if (str[i] == ' ') {
      i++;
      continue;
    }
    while (i < str.length() && str[i] != ' ') {
      tokens[j] += str[i];
      i++;
    }
    j++;
  }
  return { tokens, j };
}

bool isNum(char c) {
  return c >= '0' && c <= '9';
}

int * sepInt(string * tokens, int n) {
  int * arr = new int[n];
  for (int i = 0; i < n; i++) {
    if (isNum(tokens[i][0])) {
      arr[i] = stoi(tokens[i]);
    } else {
      arr[i] = -99;
    }
  }
  return arr;
}

int main() {
  string postfix = "25 38 * 5 4 *";
  cout << "Postfix: " << postfix << endl;

  pair < string *, int > tokens = tokenize(postfix);

  cout << "Tokens: ";
  for (int i = 0; i < tokens.second; i++) {
    cout << "'" << tokens.first[i] << "'" << ", ";
  }

  int * arr = sepInt(tokens.first, tokens.second);

  cout << endl << "Integers: ";
  for (int i = 0; i < tokens.second; i++) {
    cout << arr[i] << ", ";
  }
  return 0;
}
```

Output:

```
Postfix: 2 3 * 5 4 * +
Tokens: | 2 | 3 | * | 5 | 4 | * | + |
Result: 26

Postfix: 25 38 * 5 4 * +
Tokens: | 25 | 38 | * | 5 | 4 | * | + |
Result: 970

Postfix: 25 38 * 5 4 *
Tokens: | 25 | 38 | * | 5 | 4 | * |
Invalid postfix expression
Result: -1
```

Question 2

Question:
Write a program in C++ to find all the tokens in a given string separated by space. Also write the functions to classify a given token into operator or operands. The given string may contain integer or alphabets as operands.

Code:

```cpp
#include <iostream>
#include <string>
using namespace std;

// this comment is here to demonstrate an extremely long line length, well beyond what you should probably allow in your own code, though sometimes you'll be highlighting code you can't refactor, which is unfortunate but should be handled gracefully

pair < string *, int > tokenize(string str) {
  int n = str.length();
  string * tokens = new string[n];
  int i = 0;
  int j = 0;
  while (i < str.length()) {
    if (str[i] == ' ') {
      i++;
      continue;
    }
    while (i < str.length() && str[i] != ' ') {
      tokens[j] += str[i];
      i++;
    }
    j++;
  }
  return { tokens, j };
}

bool isNum(char c) {
  return c >= '0' && c <= '9';
}

int * sepInt(string * tokens, int n) {
  int * arr = new int[n];
  for (int i = 0; i < n; i++) {
    if (isNum(tokens[i][0])) {
      arr[i] = stoi(tokens[i]);
    } else {
      arr[i] = -99;
    }
  }
  return arr;
}

int main() {
  string postfix = "25 38 * 5 4 *";
  cout << "Postfix: " << postfix << endl;

  pair < string *, int > tokens = tokenize(postfix);

  cout << "Tokens: ";
  for (int i = 0; i < tokens.second; i++) {
    cout << "'" << tokens.first[i] << "'" << ", ";
  }

  int * arr = sepInt(tokens.first, tokens.second);

  cout << endl << "Integers: ";
  for (int i = 0; i < tokens.second; i++) {
    cout << arr[i] << ", ";
  }
  return 0;
}
```

Output:

```
Postfix: 2 3 * 5 4 * +
Tokens: | 2 | 3 | * | 5 | 4 | * | + |
Result: 26

Postfix: 25 38 * 5 4 * +
Tokens: | 25 | 38 | * | 5 | 4 | * | + |
Result: 970

Postfix: 25 38 * 5 4 *
Tokens: | 25 | 38 | * | 5 | 4 | * |
Invalid postfix expression
Result: -1
```