PPL LAB 6

SAHIL BONDRE: U18CO021

1. Write a program that reads a text file and creates another file that is identical except that every sequence of consecutive blank space is replaced by a single space.

```
#include <fstream>
#include <iostream>
#include <vector>
using namespace std;


int main() {
  string original, next;
  cout << "Enter source file name: ";
  cin >> original;
  cout << "Enter destination file name: ";
  cin >> next;
  string origText;
  ifstream MyReadFile(original);
  ofstream MyWriteFile(next);
  string word = "";
  while (getline(MyReadFile, origText)) {
    int length = origText.length();
    for (int i = 0; i < length; ++i) {
      char c = origText[i];
      if (!isspace(c)) {
```

```
        MyWriteFile << c;

      } else {

        if (i != 0 && !isspace(origText[i - 1])) {

          MyWriteFile << c;

        }

      }

    }

  }

  MyReadFile.close();

  MyWriteFile.close();

  return 0;

}
```

```
PS F:\code\github.com\godcrampy\college-notes\ppl\lab-07> .\a.exe
Enter source file name: q1_orig.txt
Enter destination file name: q1_dest.txt
PS F:\code\github.com\godcrampy\college-notes\ppl\lab-07> cat .\q1_orig.txt
This     file      has lots     of     spaces!
PS F:\code\github.com\godcrampy\college-notes\ppl\lab-07> cat .\q1_dest.txt
This file has lots of spaces!
PS F:\code\github.com\godcrampy\college-notes\ppl\lab-07>
```

2. Write a program to copy the contents of a source file student1.txt to a destination file student2.txt character by character.

```
#include <fstream>

#include <iostream>

#include <vector>

using namespace std;

int main() {

  string original, next;
```

```
  cout << "Enter source file name: ";

  cin >> original;

  cout << "Enter destination file name: ";

  cin >> next;

  string origText;

  ifstream MyReadFile(original);

  ofstream MyWriteFile(next);

  while (getline(MyReadFile, origText)) {

    int length = origText.length();

    for (int i = 0; i < length; ++i) {

      char c = origText[i];

      MyWriteFile << c;

    }

  }

  MyReadFile.close();

  MyWriteFile.close();

  return 0;

}
```

```
PS F:\code\github.com\godcrampy\college-notes\ppl\lab-07> .\a.exe
Enter source file name: q2_source.txt
Enter destination file name: q2_dest.txt
PS F:\code\github.com\godcrampy\college-notes\ppl\lab-07> cat .\q2_source.txt
hello world!
PS F:\code\github.com\godcrampy\college-notes\ppl\lab-07> cat .\q2_dest.txt
hello world!
PS F:\code\github.com\godcrampy\college-notes\ppl\lab-07>
```

3. Write a program that uses command-line argument to copy the contents of a file A.txt into another file B.txt by reversing the case of the characters. E.g. File A.txt: aBCd File B.txt: AbcD.

```
#include <cctype>
```

```cpp
#include <cstring>
#include <fstream>
#include <iostream>
#include <vector>
using namespace std;
int main(int argc, char* argv[]) {
  if (argc != 3) {
    cout << "Usage: a.out <source-file> <destination-file>\n";
    return -1;
  }
  string origText;
  ifstream MyReadFile(argv[1]);
  ofstream MyWriteFile(argv[2]);
  while (getline(MyReadFile, origText)) {
    int length = origText.length();
    for (int i = 0; i < length; ++i) {
      char c = origText[i];
      if (isupper(c)) {
        c = c + 32;
        MyWriteFile << c;
      } else if (islower(c)) {
        c = c - 32;
        MyWriteFile << c;
      } else {
        MyWriteFile << c;
      }
```

```
    }

  }

  MyReadFile.close();

  MyWriteFile.close();

  return 0;

}
```

```
PS F:\code\github.com\godcrampy\college-notes\ppl\lab-07> .\a.exe
Usage: a.out <source-file> <destination-file>
PS F:\code\github.com\godcrampy\college-notes\ppl\lab-07> .\a.exe .\q3_source.tx .\q3_dest.txt
PS F:\code\github.com\godcrampy\college-notes\ppl\lab-07> cat .\q3_source.tx
aBvD
PS F:\code\github.com\godcrampy\college-notes\ppl\lab-07> cat .\q3_dest.txt
AbVd
PS F:\code\github.com\godcrampy\college-notes\ppl\lab-07>
```

4. Write a program for swapping two values of different data types using template.

```
#include <iostream>

using namespace std;

template <class T, class U>

void swap(T& x, U& y) {

  const T tmp = x;

  x = static_cast<T>(y);

  y = static_cast<U>(tmp);

}

int main() {

  float x = 10.5;

  int y = 15;

  cout << "x: " << x << " y: " << y << endl;

  swap(x, y);
```

```
  cout << "x: " << x << " y: " << y << endl;

  return 0;

}
```

```
PS F:\code\github.com\godcrampy\college-notes\ppl\lab-07> g++ .\q4.cpp
PS F:\code\github.com\godcrampy\college-notes\ppl\lab-07> .\a.exe
x: 10.5 y: 15
x: 15 y: 10
PS F:\code\github.com\godcrampy\college-notes\ppl\lab-07>
```

5. Write a class template to represent a generic vector. Include member function to create the vector and to modify the value of a given element.

```
#include <iostream>

#include <vector>

using namespace std;

template <class T>

class Vector {

 private:

  vector<T> vec;


 public:

  void create();

  void modify(T val, int idx);

  void display();

};

template <class T>

void Vector<T>::create() {

  vector<int> m = {2, 5, 6, 7, 9};
```

```
  vec = m;
}
template <class T>
void Vector<T>::modify(T val, int idx) {
  int size = vec.size();
  if (idx >= 0 && idx < size) {
    vec[idx] = val;
  }
}
template <class T>
void Vector<T>::display() {
  int size = vec.size();
  cout << "Vector: ";
  for (int i = 0; i < size; ++i) {
    cout << vec[i] << " ";
  }
  cout << endl;
}
int main() {
  Vector<int> v;
  v.create();
  cout << "Before modifying: " << endl;
  v.display();
  v.modify(8, 2);
  cout << "After modifying: " << endl;
  v.display();
```

```cpp
  return 0;

}
```

```
PS F:\code\github.com\godcrampy\college-notes\ppl\lab-07> .\a.exe
Before modifying:
Vector: 2 5 6 7 9
After modifying:
Vector: 2 5 8 7 9
PS F:\code\github.com\godcrampy\college-notes\ppl\lab-07>
```

6. Create a generic class stack using template and implement common Push and Pop operations for different data types.

```cpp
#include <deque>

#include <iostream>

using namespace std;

template <class T>

class stack {

 private:

  deque<T> s;


 public:

  void push(T val);

  void pop();

  void display();

};

template <class T>

void stack<T>::push(T val) {

  cout << "Push operation called for value: " << val << endl;
```

```
  s.push_back(val);
}
template <class T>
void stack<T>::pop() {
  cout << "Pop operation called for value: " << s[s.size() - 1] << endl;
  s.pop_back();
}
template <class T>
void stack<T>::display() {
  cout << "Stack: ";
  for (T val : s) {
    cout << val << " ";
  }
  cout << endl;
}
int main() {
  stack<int> st;
  st.push(15);
  st.pop();
  st.push(26);
  st.push(39);
  st.display();
  return 0;
}
```

```
PS F:\code\github.com\godcrampy\college-notes\ppl\lab-07> .\a.exe
Push operation called for value: 15
Pop operation called for value: 15
Push operation called for value: 26
Push operation called for value: 39
Stack: 26 39
PS F:\code\github.com\godcrampy\college-notes\ppl\lab-07>
```