U18CO018
Shubham Shekhaliya
PPL
Lab Assignment 7

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 <bits/stdc++.h>
using namespace std;
void compress(ifstream &in, ofstream &out) {
    char c;
    bool space = false;
    while (in.get(c)) {
        if (c == ' ') {
            if (!space) {
                out.put(' ');
                space = true;
            }
        }
        Else {
            out.put(c);
            space = false;
        }
    }
}
int main() {
    string a = "in.txt";
    string b = "out.txt";
    ifstream in(a);
    ofstream out(b);
    compress(in, out);
}
```

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

```
#include <bits/stdc++.h>
using namespace std;
void copy(istream &in, ostream &out) {
    char ch;
    while (in.get(ch)) {
        out.put(ch);
    }
}
int main() {
    string a = "in.txt";
```

```
    string b = "out.txt";
    ifstream in(a);
    ofstream out(b);
    copy(in, out);
}
```

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 <bits/stdc++.h>
using namespace std;
void reverseCase(istream &in, ostream &out) {
    char c;
    while (in.get(c)) {
        if (isupper(c)) {
            c = tolower(c);
        }
        Else {
            c = toupper(c);
        }
        out.put(c);
    }
}
int main() {
    string a = "in.txt";
    string b = "out.txt";
    ifstream in(a);
    ofstream out(b);
    reverseCase(in, out);
}
```

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

```
#include <bits/stdc++.h>
using namespace std;
template <typename T>
void swap_v(T &a, T &b){
    T temp = a;
    a = b;
    b = temp;
}
int main() {
    int a = 1, b = 2;
    cout << a << " " << b << endl;
    swap_v(a, b);
    cout << a << " " << b << endl;
    float c = 1.1, d = 2.2;
```

```
    cout << c << " " << d << endl;
    swap_v(c, d);
    cout << c << " " << d << endl;
}
```

OUTPUT

```
PS E:\Git\Assignments\PPL\asgn7> ./a.exe
1 2
2 1
1.1 2.2
2.2 1.1
```

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 <bits/stdc++.h>
using namespace std;
template <class T>
class VCTR {
private:
    T *arr;
    int size;

public:
    VCTR(int n) {
        size = n;
        arr = new T[size];
    }
    void set(int i, T val) {
        arr[i] = val;
    }
    T get(int i) {
        return arr[i];
    }
};
int main() {
    int n;
    cout << "Enter the size of the vector: ";
    cin >> n;
    VCTR<int> v(n);
    int p;
    for (int i = 0; i < n; i++) {
        cout << "Enter the value of element " << i << ": ";
        cin >> p;
        v.set(i, p);
    }
```

```
    while (1) {
        cout << "1. Print vector" << endl;
        cout << "2. Modify element" << endl;
        cout << "3. Print certain element" << endl;
        cout << "4. Exit" << endl;
        int choice;
        cin >> choice;
        switch (choice) {
        case 1:
            for (int i = 0; i < n; i++) {
                cout << v.get(i) << " ";
            }
            cout << endl;
            break;
        case 2:
            int i, val;
            cout << "Enter the index of the element to be modified: ";
            cin >> i;
            cout << "Enter the new value: ";
            cin >> val;
            v.set(i, val);
            break;
        case 3:
            int index;
            cout << "Enter the index of the element to be printed: ";
            cin >> index;
            cout << v.get(index) << endl;
            break;
        case 4:
            return 0;
        default:
            cout << "Invalid choice" << endl;
        }
    }
    return 0;
}
```

OUTPUT

```
PS E:\Git\Assignments\PPL\asgn7> ./a.exe
Enter the size of the vector: 5
Enter the value of element 0: 1
Enter the value of element 1: 2
Enter the value of element 2: 3
Enter the value of element 3: 4
Enter the value of element 4: 5
1. Print vector
2. Modify element
3. Print certain element
4. Exit
1
1 2 3 4 5
1. Print vector
2. Modify element
3. Print certain element
4. Exit
2
Enter the index of the element to be modified: 3
Enter the new value: 10
1. Print vector
2. Modify element
3. Print certain element
4. Exit
1
1 2 3 10 5
1. Print vector
2. Modify element
3. Print certain element
4. Exit
3
Enter the index of the element to be printed: 3
10
```

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

```cpp
#include <bits/stdc++.h>
using namespace std;
template <class T>
class Stack {
private:
    int top;
    int capacity;
    T *array;

public:
    Stack(int capacity) {
        this->capacity = capacity;
```

```
        top = -1;
        array = new T[capacity];
    }
    bool isFull() {
        return top == capacity - 1;
    }
    bool isEmpty() {
        return top == -1;
    }
    void push(T data) {
        if (isFull()) {
            cout << "Stack is full" << endl;
            return;
        }
        array[++top] = data;
    }
    T pop() {
        if (isEmpty()) {
            cout << "Stack is empty" << endl;
            return -1;
        }
        return array[top--];
    }
    T peek() {
        if (isEmpty()) {
            cout << "Stack is empty" << endl;
            return -1;
        }
        return array[top];
    }
};
int main() {
    int n;
    cout << "Enter the number of elements in the stack: ";
    cin >> n;
    Stack<int> s(n);
    while (1) {
        cout << "1. PUSH" << endl;
        cout << "2. POP" << endl;
        cout << "3. PEEK" << endl;
        cout << "4. EXIT" << endl;
        int choice;
        cin >> choice;
        switch (choice) {
        case 1: {
            int data;
            cout << "Enter the data to be pushed: ";
            cin >> data;
```

```
            s.push(data);
            break;
        }
        case 2: {
            cout << "Popped element: " << s.pop() << endl;
            break;
        }
        case 3: {
            cout << "Peeked element: " << s.peek() << endl;
            break;
        }
        case 4: {
            return 0;
        }
        default: {
            cout << "Invalid choice" << endl;
        }
        }
    }
    return 0;
}
```