```cpp
// Frank Bodholdt Jacobsen
// Studienr. 197600001

// Opgave 1A

// Document.h
#include <string>
using namespace std;

class Document
{
public:
    // Constructor decucted from usage in Jpg class
    Document(string, bool);
    // Alle basisklasser bør have virtual destructor
    virtual ~Document();

    void setName(std::string);
    string getName() const;
    void setPriority(bool);
    bool getPriority() const;
    virtual void print() const = 0;

private:
    std::string name_;
    bool priority_;
};

// Document.cpp
#include "Document.h"

Document::Document(string name, bool prio)
    : name_(name), priority_(prio)
{

}

Document::~Document()
{

}

void Document::setName(string name)
{
    name_ = name;
}

string Document::getName() const
{
    return name_;
}

void Document::setPriority(bool prio)
{
    priority_ = prio;
}

bool Document::getPriority() const
{
    return priority_;
}

// Opgave 1B

```

```
65  // Txt.h
66  #pragma once
67  #include "Document.h"
68  #include <iostream>
69
70  class Txt : public Document
71  {
72  public:
73      Txt(string, bool = false);
74      ~Txt();
75      void setText(char *);
76      const char* getText() const;
77      void print() const;
78
79      // Ekstra metoder opgave 1F
80      const Txt & operator=(const Txt &);
81      Txt(const Txt &);
82
83  private:
84      char * text_;
85
86  };
87
88  // Txt.cpp
89  #include "Txt.h"
90  #include <iostream>
91
92  Txt::Txt(string name, bool priority)
93      : Document(name + ".txt", priority)
94  {
95      text_ = new char[1]{ '\0' };
96      *text_ = '\0';
97      text_[0] = '\0';
98  }
99
100 Txt::~Txt()
101 {
102     delete[] text_;
103     cout << "Memory for text_ is deallocated" << endl;
104 }
105
106 void Txt::print() const
107 {
108     cout << endl << (getPriority() == 1 ? "High: " : "Low:  ");
109     cout << getName() << endl << endl;
110     cout << text_ << endl << endl;
111 }
112
113 const char * Txt::getText() const
114 {
115     return text_;
116 }
117
118 // Opgave 1C - del af txt.cpp
119 void Txt::setText(char * newText)
120 {
121     if (strlen(text_) != strlen(newText))
122     {
123         delete[] text_;
124         text_ = new char[strlen(newText) + 1];
125     }
126
127     strcpy(text_, newText);
128 }
```

```
129
130 // Opgave 1D
131 #include "Txt.h"
132 #include <iostream>
133
134 using namespace std;
135
136 int main()
137 {
138     Document *docPtr;
139     docPtr = new Txt("Eksempel");
140
141     delete docPtr;
142     docPtr = nullptr;
143
144     Txt myText("MyTxt");
145     myText.setText("Invitation til pyjamasparty");
146
147     //myText.print();
148
149     cout << myText;
150
151     return 0;
152 }
153
154 // Opgave 1E
155 // Del af txt.h
156 ostream & operator<<(ostream &os, const Txt &);
157
158 // Del af txt.cpp
159 ostream & operator<<(ostream & os, const Txt &t)
160 {
161     os << endl << (t.getPriority() == 1 ? "High: " : "Low:  ");
162     os << t.getName() << endl << endl;
163     os << t.getText() << endl << endl;
164
165     return os;
166 }
167
168 // Opgave 1F
169 // To ekstra metoder i class Txt
170 class Txt : public Document
171 {
172 public:
173
174     // Ekstra metoder opgave 1F
175     const Txt & operator=(const Txt &);
176     Txt(const Txt &);
177
178
179 };
180
181
182
183 // Opgave 2A
184
185 // Printer.h
186 #pragma once
187
188 #include "Document.h"
189 #include <deque>
190 using namespace std;
191
192 class Printer
```

```
193 {
194 public:
195     void addToQueue(Document *);
196     void printNextDocument();
197 
198     void showQueue() const;
199 
200 private:
201     deque<Document *> printerQueue_;
202 };
203 
204 // Printer.cpp
205 #include "Printer.h"
206 
207 #include <iostream>
208 using namespace std;
209 
210 void Printer::addToQueue(Document * d)
211 {
212     if (d->getPriority())
213     {
214         // High priority, put in front
215         printerQueue_.push_front(d);
216     }
217     else
218     {
219         printerQueue_.push_back(d);
220     }
221 }
222 
223 void Printer::printNextDocument()
224 {
225     if (printerQueue_.empty())
226     {
227         cout << "Print queue is empty\n\n";
228     }
229     else
230     {
231         printerQueue_.front()->print();
232         printerQueue_.pop_front();
233     }
234 }
235 
236 // Opgave 2B
237 // Tilføjet til main.cpp
238 // tilføj kode her jvf. opgave 2B
239 Printer printer;
240 
241 printer.addToQueue(&text1);
242 printer.addToQueue(&pic1);
243 printer.addToQueue(&text2);
244 printer.addToQueue(&text3);
245 printer.addToQueue(&pic2);
246 
247 for (int i = 0; i < 6; i++)
248 {
249     printer.printNextDocument();
250 }
251 
252 // Opgave 2C
253 // Del af printer.cpp
254 void Printer::showQueue() const
255 {
256     deque<Document *>::const_iterator it;
```

```
257     for (it = printerQueue_.begin(); it != printerQueue_.end(); it++)
258     {
259         const Document * dPtr = *it;
260
261         if (dPtr->getPriority())
262         {
263             cout << "High: ";
264         }
265         else
266         {
267             cout << "Low:  ";
268         }
269
270         cout << dPtr->getName() << endl;
271     }
272 }
273
274
```