

# cs225f20-b.sg

[Dashboard](#) / [My courses](#) / [cs225f20-b.sg](#) / [16 November - 22 November](#) / [Assignment 6 upload](#)

- Description
- Submission view

## Grade

Reviewed on Thursday, 10 December 2020, 9:29 AM by Slawomir Grzegorz Wlodkowski  
**grade:** A

**Assessment report**[\[-\]](#)  
[\[+\]](#) **Summary of tests**

Submitted on Thursday, 26 November 2020, 8:27 PM ([Download](#))

allocator.cpp

```
1  ▾ /* Start Header
2  *****/
3  ▾ /*!
4  \file  allocator.cpp
5  \author Goh Wei Zhe, weizhe.goh, 44000119
6  \par    email: weizhe.goh@digipen.edu
7  \date   November 26, 2020
8  \brief  Contains definition of class allocator
9
10 Copyright (C) 2020 DigiPen Institute of Technology.
11 Reproduction or disclosure of this file or its contents without the
12 prior written consent of DigiPen Institute of Technology is prohibited.
13 */
14 ▾ /* End Header
15 *****/
16
17 #include "allocator.h"
18 #include <iostream>
19
20 unsigned char* cs225::allocator::allocate(size_t count)
21 ▾ {
22     size_t size = (count + CHAR_BIT - 1) / CHAR_BIT;
23     unsigned char* bitset = new unsigned char[size] {0};
24
25     std::cout << "Allocate " << size << " elements." << std::endl;
26     return bitset;
27 }
28
29 void cs225::allocator::deallocate(unsigned char* ptr, size_t count)
30 ▾ {
31     size_t size = (count + CHAR_BIT - 1) / CHAR_BIT;
32     delete[] ptr;
33
34     std::cout << "Deallocate " << size << " elements." << std::endl;
35 }
```

allocator.h

```
1  ▾ /* Start Header
2  *****/
3  ▾ /*!
4  \file  allocator.h
5  \author Goh Wei Zhe, weizhe.goh, 440000119
6  \par    email: weizhe.goh\@digipen.edu
7  \date   November 26, 2020
8  \brief  Contains declaration of class allocator
9
10 Copyright (C) 2020 DigiPen Institute of Technology.
11 Reproduction or disclosure of this file or its contents without the
12 prior written consent of DigiPen Institute of Technology is prohibited.
13 */
14 ▾ /* End Header
15 *****/
16
17 #ifndef ALLOCATOR_H
18 #define ALLOCATOR_H
19
20 #include <memory>
21 #include <climits>
22
23 namespace cs225
24 ▾ {
25     class allocator
26     ▾ {
27     public:
28         static unsigned char* allocate(size_t count);
29         static void deallocate(unsigned char* ptr, size_t count);
30     };
31 }
32
33 #endif
```

bitset.h

```
1  ▾ /* Start Header
2  *****/
3  ▾ /*!
4  \file  bitset.h
5  \author Goh Wei Zhe, weizhe.goh, 440000119
6  \par    email: weizhe.goh\@digipen.edu
7  \date   November 26, 2020
8  \brief  Contains declaration of member functions of class bitset
9
10 Copyright (C) 2020 DigiPen Institute of Technology.
11 Reproduction or disclosure of this file or its contents without the
12 prior written consent of DigiPen Institute of Technology is prohibited.
13 */
14 ▾ /* End Header
15 *****/
16
17 #ifndef BITSET_H
18 #define BITSET_H
19
20 #include <bitset>
21
22 namespace cs225
23 ▾ {
24     template<std::size_t N, typename Allocator>
25     class bitset
26     ▾ {
27     public:
28
29         unsigned char* _bitset;
30
31         bitset();
32
33         std::size_t count() const;
34         std::size_t size() const;
35
36         bitset<N, Allocator>& set(std::size_t pos, bool value = true);
37         bitset<N, Allocator>& reset(std::size_t pos);
38         bitset<N, Allocator>& flip(std::size_t pos);
39
40         bool test(std::size_t pos) const;
41         bool operator[](std::size_t pos) const;
42
43         std::string to_string(char c1 = '0' , char c2 = '1') const;
44
45         ~bitset();
46     };
47 }
48
49 #include "bitset.hpp"
50
51 #endif
```

bitset.hpp

```

1  /* Start Header
2  *****/
3  /*!
4  \file   bitset.hpp
5  \author Goh Wei Zhe, weizhe.goh, 440000119
6  \par    email: weizhe.goh@digipen.edu
7  \date   November 26, 2020
8  \brief  Contains definition of member functions of class bitset
9
10 Copyright (C) 2020 DigiPen Institute of Technology.
11 Reproduction or disclosure of this file or its contents without the
12 prior written consent of DigiPen Institute of Technology is prohibited.
13 */
14 /* End Header
15 *****/
16
17 #ifndef BITSET_HPP
18 #define BITSET_HPP
19
20 #include "bitset.h"
21
22 template<std::size_t N, typename Allocator>
23 cs225::bitset<N, Allocator>::bitset()
24 {
25     _bitset = cs225::allocator::allocate(N);
26 }
27
28 template<std::size_t N, typename Allocator>
29 cs225::bitset<N, Allocator>& cs225::bitset<N, Allocator>
30 ::set(size_t pos, bool value)
31 {
32     if (value)
33         _bitset[pos / CHAR_BIT] |=
34         static_cast<unsigned char>(1U << pos % CHAR_BIT);
35     else
36         _bitset[pos / CHAR_BIT] &=
37         static_cast<unsigned char>(~(1U << pos % CHAR_BIT));
38
39     return *this;
40 }
41
42 template<std::size_t N, typename Allocator>
43 cs225::bitset<N, Allocator>& cs225::bitset<N, Allocator>::reset(size_t pos)
44 {
45     _bitset[pos / CHAR_BIT] &=
46     static_cast<unsigned char>(~(1U << pos % CHAR_BIT));
47
48     return *this;
49 }
50
51 template<std::size_t N, typename Allocator>
52 cs225::bitset<N, Allocator>& cs225::bitset<N, Allocator>::flip(size_t pos)
53 {
54     _bitset[pos / CHAR_BIT] ^=
55     static_cast<unsigned char>(1U << (pos % CHAR_BIT));
56
57     return *this;
58 }
59
60 template<std::size_t N, typename Allocator>
61 size_t cs225::bitset<N, Allocator>::count() const
62 {
63     size_t count = 0;
64
65     for (size_t i = N; i > 0;)
66         (bitset<N, Allocator>::test(--i) ? ++count : count);
67
68     return count;
69 }
70
71 template<std::size_t N, typename Allocator>
72 std::string cs225::bitset<N, Allocator>::to_string(char c1, char c2) const
73 {
74     std::string s;
75
76     char zero = static_cast<unsigned char>(c1);
77     char one = static_cast<unsigned char>(c2);
78
79     for (size_t i = N; i > 0;)
80         s.push_back(bitset<N, Allocator>::test(--i) ? one : zero);
81
82     return s;
83 }
84
85 template<std::size_t N, typename Allocator>
86 size_t cs225::bitset<N, Allocator>::size() const
87 {
88     return N;
89 }
90
91 template<std::size_t N, typename Allocator>
92 bool cs225::bitset<N, Allocator>::test(size_t pos) const
93 {
94     if (N <= pos)
95         throw std::out_of_range("Out of range");
96
97     return (_bitset[pos / CHAR_BIT] & (1U << pos % CHAR_BIT)) != 0;
98 }
99
100 template<std::size_t N, typename Allocator>
101 bool cs225::bitset<N, Allocator>::operator[](size_t pos) const
102 {
103     return (_bitset[pos / CHAR_BIT] & (1U << pos % CHAR_BIT)) != 0;
104 }
105
106 template<std::size_t N, typename Allocator>
107 cs225::bitset<N, Allocator>::~~bitset()

```

```
108 {
109     cs225::allocator::deallocate(_bitset, N);
110 }
111
```

bitset\_tep.h

```
1  /* Start Header
2  *****/
3  /*!
4  \file   bitset_tep.h
5  \author Goh Wei Zhe, weizhe.goh, 44000119
6  \par    email: weizhe.goh@digipen.edu
7  \date   November 26, 2020
8  \brief  Contains declaration of class bitset_tep
9
10 Copyright (C) 2020 DigiPen Institute of Technology.
11 Reproduction or disclosure of this file or its contents without the
12 prior written consent of DigiPen Institute of Technology is prohibited.
13 */
14 /* End Header
15 *****/
16
17 #ifndef BITSET_TEP_H
18 #define BITSET_TEP_H
19
20 #include <string>
21 #include <iostream>
22 #include <memory>
23
24 namespace cs225
25 {
26     class bitset_tep
27     {
28     public:
29         struct IConcept
30         {
31             virtual size_t size() const = 0;
32             virtual size_t count() const = 0;
33
34             virtual void set(size_t, bool = true) = 0;
35             virtual void flip(size_t) = 0;
36             virtual void reset(size_t) = 0;
37
38             virtual bool test(size_t) const = 0;
39             virtual bool operator[](std::size_t pos) const = 0;
40
41             virtual std::string to_string(char = '0', char = '1') const = 0;
42             virtual ~IConcept() = default;
43         };
44
45         template<typename T>
46         class Model : public IConcept
47         {
48         public:
49             T _instance;
50
51             template<typename... Args>
52             Model(Args&&... args);
53
54             virtual size_t size() const override;
55             virtual size_t count() const override;
56
57             virtual void set(size_t pos, bool value = true) override;
58             virtual void flip(size_t pos) override;
59             virtual void reset(size_t pos) override;
60
61             virtual bool test(size_t pos) const override;
62             virtual bool operator[](std::size_t pos) const override;
63
64             virtual std::string to_string(char = '0', char = '1') const override;
65         };
66
67         std::unique_ptr<IConcept> _concept;
68         bitset_tep(std::unique_ptr<IConcept> concept);
69
70     public:
71         std::size_t count() const;
72         std::size_t size() const;
73
74         void set(size_t pos, bool value = true);
75         void flip(size_t pos) const;
76         void reset(size_t pos) const;
77
78         bool test(size_t pos) const;
79         bool operator[](std::size_t pos) const;
80
81         std::string to_string(char c1 = '0', char c2 = '1') const;
82
83         template<typename T, typename... Args>
84         static bitset_tep create(Args&&... args);
85     };
86 }
87
88 #include "bitset_tep.hpp"
89
90 #endif // !_BITSET_TEP_H
91
92
```

bitset\_tep.hpp

```
1  /* Start Header
2  *****/
3  /*!
4  \file   bitset_tep.hpp
5  \author Goh Wei Zhe, weizhe.goh, 440000119
6  \par    email: weizhe.goh\@digipen.edu
7  \date   November 26, 2020
8  \brief  Contains definition of templated alias of class bitset_tep
9
10 Copyright (C) 2020 DigiPen Institute of Technology.
11 Reproduction or disclosure of this file or its contents without the
12 prior written consent of DigiPen Institute of Technology is prohibited.
13 */
14 /* End Header
15 *****/
16 #include <iostream>
17
18 #ifndef BITSET_TEP_HPP
19 #define BITSET_TEP_HPP
20
21 #include "bitset_tep.h"
22 #include <iostream>
23
24 template<typename T> template <typename... Args>
25 cs225::bitset_tep::Model<T>::Model(Args&&... args)
26 : _instance{std::forward<Args>(args)...}{}
27
28 template<typename T>
29 size_t cs225::bitset_tep::Model<T>::size() const
30 {
31     return _instance.size();
32 }
33
34 template<typename T>
35 size_t cs225::bitset_tep::Model<T>::count() const
36 {
37     return _instance.count();
38 }
39
40 template<typename T>
41 void cs225::bitset_tep::Model<T>::set(size_t pos, bool value)
42 {
43     _instance.set(pos, value);
44 }
45
46 template<typename T>
47 void cs225::bitset_tep::Model<T>::flip(size_t pos)
48 {
49     _instance.flip(pos);
50 }
51
52 template<typename T>
53 void cs225::bitset_tep::Model<T>::reset(size_t pos)
54 {
55     _instance.reset(pos);
56 }
57
58 template<typename T>
59 bool cs225::bitset_tep::Model<T>::test(size_t pos) const
60 {
61     return _instance.test(pos);
62 }
63
64 template<typename T>
65 std::string cs225::bitset_tep::Model<T>::to_string(char c1, char c2) const
66 {
67     return _instance.to_string(c1, c2);
68 }
69
70 template<typename T>
71 bool cs225::bitset_tep::Model<T>::operator[](size_t pos) const
72 {
73     return _instance.operator[](pos);
74 }
75
76 template<typename T, typename... Args>
77 cs225::bitset_tep cs225::bitset_tep::create(Args&&... args)
78 {
79     return bitset_tep(std::make_unique<Model<T, Args...>>
80         (std::forward<Args>(args)...));
81 }
82
83 #endif // !BITSET_TEP_HPP
84
```

bitset\_tep.cpp

```
1  ▾ /* Start Header
2  *****
3  ▾ */
4  \file  bitset_tep.cpp
5  \author Goh Wei Zhe, weizhe.goh, 440000119
6  \par    email: weizhe.goh\@digipen.edu
7  \date   November 26, 2020
8  \brief  Contains definition of non-templated alias of class bitset_tep
9
10 Copyright (C) 2020 DigiPen Institute of Technology.
11 Reproduction or disclosure of this file or its contents without the
12 prior written consent of DigiPen Institute of Technology is prohibited.
13 */
14 ▾ /* End Header
15 *****
16 #include <iostream>
17
18 #include "bitset_tep.h"
19
20 cs225::bitset_tep::bitset_tep(std::unique_ptr<cs225::bitset_tep::IConcept>concept)
21     : _concept{std::move(concept) } {}
22
23 size_t cs225::bitset_tep::count() const
24 ▾ {
25     return _concept->count();
26 }
27
28 size_t cs225::bitset_tep::size() const
29 ▾ {
30     return _concept->size();
31 }
32
33 void cs225::bitset_tep::set(size_t pos, bool value)
34 ▾ {
35     _concept->set(pos, value);
36 }
37
38 void cs225::bitset_tep::flip(size_t pos) const
39 ▾ {
40     _concept->flip(pos);
41 }
42
43 void cs225::bitset_tep::reset(size_t pos) const
44 ▾ {
45     _concept->reset(pos);
46 }
47
48 bool cs225::bitset_tep::test(size_t pos) const
49 ▾ {
50     return _concept->test(pos);
51 }
52
53 bool cs225::bitset_tep::operator[](size_t pos) const
54 ▾ {
55     return _concept->operator[](pos);
56 }
57
58 std::string cs225::bitset_tep::to_string(char c1, char c2) const
59 ▾ {
60     return _concept->to_string(c1, c2);
61 }
```

[VPL](#)

◀ Assignment 6 files

Jump to...

⌵

Attendance cs225f20-b.sg Tuesday  
24/11/2020 1:30pm-3:10pm ▶

You are logged in as [Wei Zhe GOH](#) ([Log out](#))  
[cs225f20-b.sg](#)  
[Data retention summary](#)  
[Get the mobile app](#)