

```
1 // Implements a list of unique numbers using an array of fixed length
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Prompt for number of numbers
9     int capacity;
10    do
11    {
12        capacity = get_int("capacity: ");
13    }
14    while (capacity < 1);
15
16    // Memory for numbers
17    int numbers[capacity];
18
19    // Prompt for numbers
20    int size = 0;
21    while (size < capacity)
22    {
23        // Prompt for number
24        int number = get_int("number: ");
25
26        // Check whether number is already in list
27        bool found = false;
28        for (int i = 0; i < size; i++)
29        {
30            if (numbers[i] == number)
31            {
32                found = true;
33                break;
34            }
35        }
36
37        // If number not found in list, add to list
38        if (!found)
39        {
40            numbers[size] = number;
41            size++;
42        }
43    }
44
45    // Print numbers
```

---

```
46     for (int i = 0; i < size; i++)
47     {
48         printf("%i\n", numbers[i]);
49     }
50 }
```

```
1 // Implements a list of unique numbers using an array of dynamic length
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Memory for numbers
9     int *numbers = NULL;
10    int capacity = 0;
11
12    // Prompt for numbers (until EOF)
13    int size = 0;
14    while (true)
15    {
16        // Prompt for number
17        int number = get_int("number: ");
18
19        // Check for EOF
20        if (number == INT_MAX)
21        {
22            break;
23        }
24
25        // Check whether number is already in list
26        bool found = false;
27        for (int i = 0; i < size; i++)
28        {
29            if (numbers[i] == number)
30            {
31                found = true;
32                break;
33            }
34        }
35
36        // If number not found in list, add to list
37        if (!found)
38        {
39            // Check whether enough space for number
40            if (size == capacity)
41            {
42                // Allocate space for number
43                int *tmp = realloc(numbers, sizeof(int) * (size + 1));
44                if (!tmp)
45                {
```

```
46         if (numbers)
47         {
48             free(numbers);
49         }
50         return 1;
51     }
52     numbers = tmp;
53     capacity++;
54 }
55
56     // Add number to list
57     numbers[size] = number;
58     size++;
59 }
60 }
61
62 // Print numbers
63 printf("\n");
64 for (int i = 0; i < size; i++)
65 {
66     printf("%i\n", numbers[i]);
67 }
68
69 // Free memory
70 if (numbers)
71 {
72     free(numbers);
73 }
74 }
```

```
1 // Implements a list of unique numbers using a linked list
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 typedef struct node
7 {
8     int number;
9     struct node *next;
10 }
11 node;
12
13 int main(void)
14 {
15     // Memory for numbers
16     node *numbers = NULL;
17
18     // Prompt for numbers (until EOF)
19     while (true)
20     {
21         // Prompt for number
22         int number = get_int("number: ");
23
24         // Check for EOF
25         if (number == INT_MAX)
26         {
27             break;
28         }
29
30         // Check whether number is already in list
31         bool found = false;
32         for (node *ptr = numbers; ptr != NULL; ptr = ptr->next)
33         {
34             if (ptr->number == number)
35             {
36                 found = true;
37                 break;
38             }
39         }
40
41         // If number not found in list, add to list
42         if (!found)
43         {
44             // Allocate space for number
45             node *n = malloc(sizeof(node));
```

```
46     if (!n)
47     {
48         return 1;
49     }
50
51     // Add number to list
52     n->number = number;
53     n->next = NULL;
54     if (numbers)
55     {
56         for (node *ptr = numbers; ptr != NULL; ptr = ptr->next)
57         {
58             if (!ptr->next)
59             {
60                 ptr->next = n;
61                 break;
62             }
63         }
64     }
65     else
66     {
67         numbers = n;
68     }
69 }
70
71
72 // Print numbers
73 printf("\n");
74 for (node *ptr = numbers; ptr != NULL; ptr = ptr->next)
75 {
76     printf("%i\n", ptr->number);
77 }
78
79 // Free memory
80 node *ptr = numbers;
81 while (ptr != NULL)
82 {
83     node *next = ptr->next;
84     free(ptr);
85     ptr = next;
86 }
87 }
```

---

```
1 // http://valgrind.org/docs/manual/quick-start.html#quick-start.prepare
2
3 #include <stdlib.h>
4
5 void f(void)
6 {
7     int *x = malloc(10 * sizeof(int));
8     x[10] = 0;
9 }
10
11 int main(void)
12 {
13     f();
14     return 0;
15 }
```

```
1 // Demonstrates structs
2
3 #include <cs50.h>
4 #include <stdio.h>
5 #include <string.h>
6
7 #include "struct.h"
8
9 int main(void)
10 {
11     // allocate space for students
12     int enrollment = get_int("enrollment: ");
13     student students[enrollment];
14
15     // prompt for students' names and dorms
16     for (int i = 0; i < enrollment; i++)
17     {
18         students[i].name = get_string("name: ");
19         students[i].dorm = get_string("dorm: ");
20     }
21
22     // print students' names and dorms
23     for (int i = 0; i < enrollment; i++)
24     {
25         printf("%s is in %s.\n", students[i].name, students[i].dorm);
26     }
27 }
```



```
1 // Demonstrates file I/O
2
3 #include <cs50.h>
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <string.h>
7
8 #include "struct.h"
9
10 int main(void)
11 {
12     // allocate memory for students
13     int enrollment = get_int("enrollment: ");
14     student students[enrollment];
15
16     // prompt for students' names and dorms
17     for (int i = 0; i < enrollment; i++)
18     {
19         students[i].name = get_string("name: ");
20         students[i].dorm = get_string("dorm: ");
21     }
22
23     // save students to disk
24     FILE *file = fopen("students.csv", "w");
25     if (file)
26     {
27         for (int i = 0; i < enrollment; i++)
28         {
29             fprintf(file, "%s,%s\n", students[i].name, students[i].dorm);
30         }
31         fclose(file);
32     }
33 }
```

---

```
1 // Represents a student
2
3 typedef struct
4 {
5     char *name;
6     char *dorm;
7 }
8 student;
```