

CIS 1200 Midterm II March 28, 2025

Name: _____

PennKey (penn login, e.g., sweirich): _____

PennID (the “numbers”, e.g., 12001200): _____

I certify that I have complied with the University of Pennsylvania’s Code of Academic Integrity in completing this examination.

Signature: _____ Date: _____

- Please wait to begin the exam until you are told it is time for everyone to start.
- When you begin, start by writing your PennKey at the bottom of all the odd-numbered pages in the rest of the exam.
- There are 120 total points. The time for the exam is 60 minutes.
- You may use a single, handwritten sheet of notes during the exam.
- For coding problems, aim for accurate syntax, but we will not grade your code for indentation, spacing, etc.
- There are 15 pages in the exam and an appendix for your reference. Do not write any answers in the appendix as they will not be graded.
- Do not spend too much time on any one question. Be sure to recheck all of your answers.
- If you need extra space for an answer, you may use the scratch page at the end of the exam; make sure to clearly indicate that you have done this in the normal answer space for the problem.
- Good luck!

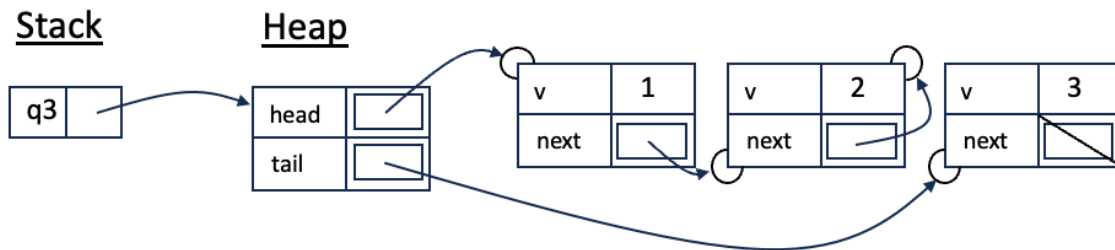
1. Programming with Linked Queues (44 points total)

Recall the type definitions of the queue data structure:

```
type 'a qnode = { v: 'a;  
                  mutable next: 'a qnode option }
```

```
type 'a queue = { mutable head: 'a qnode option;  
                  mutable tail: 'a qnode option }
```

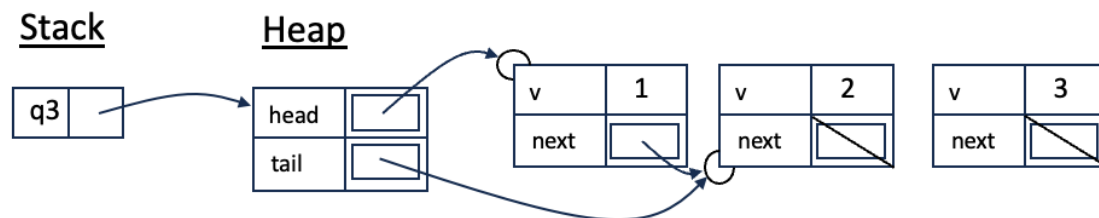
We define an *almost-queue* to be a queue that contains a cycle. Here is an example:



The goal for this problem is to define a function, called `remove_cycle`, that can convert an *almost-queue* to be a valid queue by truncating the queue at the problematic spot. (When given a *valid* queue, `remove_cycle` should not modify its argument.)

```
val remove_cycle : 'a queue -> unit
```

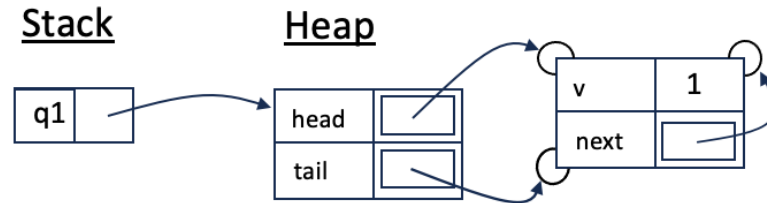
After a call to `remove_cycle`, the first node in the queue that refers to itself or to any earlier node should be the *last* node in the queue. For example, calling `remove_cycle` with the queue above could result in the following ASM configuration.



1.1 (4 points) Recall the representation invariant for linked queues:

- (1) `head` and `tail` are both tagged `None` or both tagged `Some`, and
- (2) if `head = Some h` and `tail = Some t`, then
 - (2a) `t` is reachable from `h` by following `next` pointers
 - (2b) `t.next = None`

What part of this invariant does the *almost-queue* `q1` (shown below) break, if any? (Select “Valid” if no invariants are broken.)



- ☐ (1) ☐ (2a) ☐ (2b) ☐ This queue is valid.

1.2 (10 points) Define the function `make_q1` that, when called, returns a queue that looks like `q1`. In other words, placing `let q1 = make_q1 ()` on the workspace should produce the stack and heap diagram shown above.

let `make_q1 () : int queue =`

Consider the function `contains_qn`, shown below. When given a `'a qnode` and `'a qnode list`, this function returns `true` if the `'a qnode` is present within the list and `false` otherwise.

```
let rec contains_qn (e : 'a qnode) (l : 'a qnode list) : bool =  
  begin match l with  
  | [] -> false  
  | hd :: tl -> if e == hd then true else contains_qn e tl  
  end
```

1.3 (4 points) Complete the following test case for `contains_qn`:

```
let test () : bool =  
  let qn1 = { v = 1 ; next = None } in  
  let qn2 = qn1 in  
  let qn3 = { v = 3 ; next = Some qn1 } in
```

`contains_qn qn1 [qn2 ; qn3] =`

1.4 (5 points) Is `contains_qn` tail recursive?

☐ Tail Recursive

☐ Not Tail Recursive

1.5 (5 points) Now consider a mystery function that works with queues. Don't try to understand what this function does. Is this function tail recursive?

```
let mystery (q : bool queue) : bool =  
  let rec loop (qno : bool qnode option) : bool =  
    begin match qno with  
    | None -> true  
    | Some qn -> loop qn.next && qn.v  
    end in  
  loop q.head
```

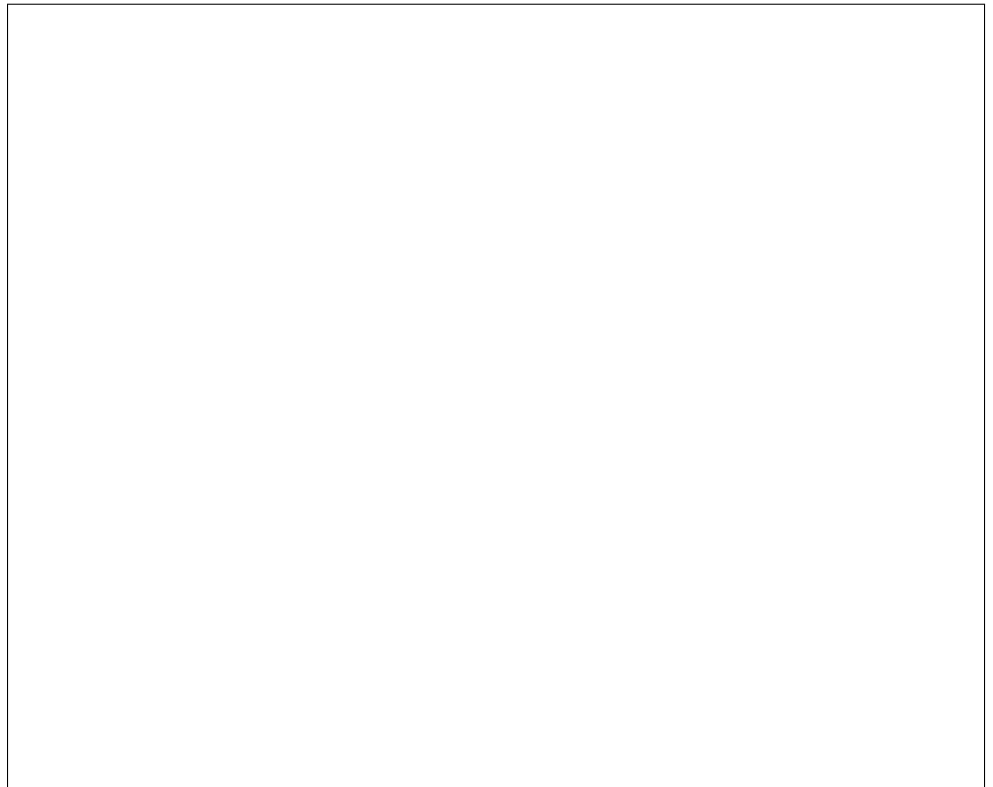
☐ Tail Recursive

☐ Not Tail Recursive

1.6 (16 points) Now complete the implementation of `remove_cycle`. We have given you a start by defining the `loop` that traverses each node in the queue. This loop maintains a list of `qnodes` that have been previously encountered. If you encounter a `qnode` that is already contained in the list, then you have found a cycle!

In your solution, you may use `contains_qn` but no other library functions.

```
let remove_cycle (q: 'a queue) : unit =  
  let rec loop (qno : 'a qnode option) (l: 'a qnode list) : unit =  
    begin match qno with  
      | None -> ()  
      | Some qn ->  
        begin match qn.next with  
          | None -> ()  
          | Some next_qn ->
```



```
    end  
  end in
```

```
loop
```

2. GUI library and Reactive Programming (32 points total)

Suppose we would like to add a simple clock (a circle with an “hour hand” indicating the current time) to the widget library from homework 5. For example, Figure 1 shows clocks set at two different times: noon and five o’clock.

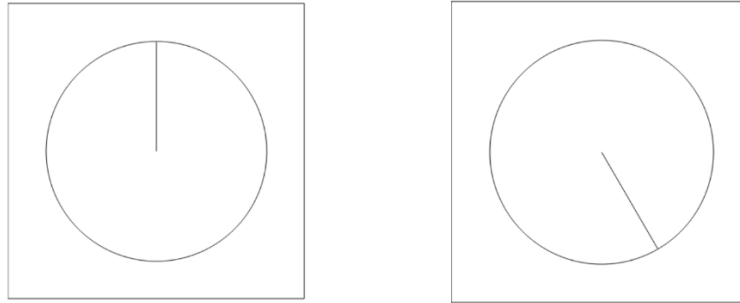


Figure 1: Clock widget at 12 o’clock (left) and 5 o’clock (right)

To do so, we will add a “clock constructor,” with the following type to `widget.mli`:

```
val clock : int -> widget * notifier_controller * int value_controller
```

When given an initial time (a number between 1 and 12, inclusive) this function creates a clock widget plus a way to add event listeners to the clock widget (using a notifier controller) and a way to add change listeners for the clock’s time (using a value controller).

First, let’s make sure that you understand the widget library. The following problem refers to the implementation of the `notifier` widget, reprinted in Appendix C.

2.1 (4 points) What is the type of the anonymous function defined on lines 12–14:

2.2 (4 points) When allocated in the heap, the closure for the anonymous function defined on lines 12–14, must include copies of the stack bindings for which of the following:

☐ `w` ☐ `listeners` ☐ `contents` ☐ `List.iter` ☐ `g` ☐ `e`

2.3 (14 points) Implement `clock`. For reference, the interface for the `Gctx` library appears in Appendix A and you may assume that `;; open Gctx` appears at the top of the module.

You should draw the clock as a 150×150 pixel ellipse centered in a 400×400 pixel canvas as the “clock face”, plus a line from the center to the edge as the “clock hand”. In your code, you may use `find_endpoint`, declared below. (You not need to implement this function.)

```
(* Calculate the correct position of the end of the "clock hand" at
   a given time. This position is a point on the circumference
   of a radius 150 circle centered at location (200,200). *)
```

```
let find_endpoint (time: int) : Gctx.position = ...
```

```
(** Create a clock widget *)
```

```
let clock (init : int) :
    widget * notifier_controller * int value_controller =
```

```
    (* value controller for the time *)
```

```
    let time = make_controller (init mod 12) in
```

```
    (* draw the clock face and clock hand *)
```

```
    let repaint (g : Gctx.gctx) : unit =
```

```
        let c : Gctx.position = (200,200) in (* center *)
```

```
        let r : int = 150 in (* radius *)
```

```
in
```

```
    (* create the clock widget and its notifier controller *)
```

```
    let clock_w, nc =
```

```
in
```

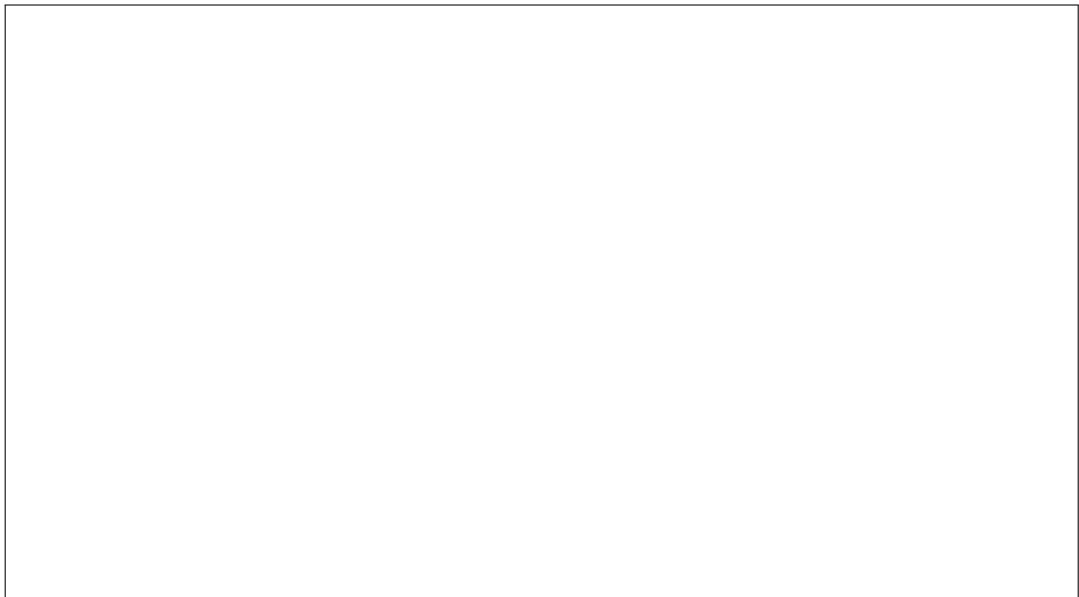
```
    (* return the appropriate values *)
```

Now, consider how a client might *use* the clock in an application. For reference, an excerpt of the `Widget` library interface appears in Appendix B. You may assume that `;; open Widget` appears at the top of the module.

- 2.4 (10 points) Say we would like to simulate an alarm clock. For example, let's create a label and put it next to the clock widget:

```
let clock_w, nc, vc = clock 12
let label_w, lc = label "Zzzzz"
let top = hpair clock_w (hpair (space (10,10)) label_w)
```

Use the clock's value controller below to make it so that when the time reaches 5, the label's string changes from "Zzzzz" to "Wake up". (You do not need to change the string at any other time.)



3. Java arrays and strings (20 points total)

The following questions refer to Java's `String` class. An excerpt of the documentation for this class is shown in Appendix D.

3.1 (4 points) What happens when the following code is executed in Java? (Select one.)

```
String[] s = {};  
int result = s.length;
```

- ☐ The `s.length` expression returns 0.
- ☐ The expression `s.length` is skipped.
- ☐ A `NullPointerException` is thrown.
- ☐ The program does not compile.
- ☐ None of the above.

3.2 (4 points) What happens when the following code is executed in Java? (Select one.)

```
String s1 = "Java";  
String s2 = new String("Java");  
System.out.println(s1.equals(s2));
```

- ☐ The program prints **true**.
- ☐ The program prints **false**.
- ☐ The program prints `null`.
- ☐ The program does not compile.
- ☐ None of the above.

3.3 (4 points) What happens when the following code is executed in Java? (Select one.)

```
String s1 = "Java";  
char[] a1 = s1.toCharArray();  
char[] a2 = { 'J', 'a', 'v', 'a' };  
System.out.println(a1 == a2);
```

- ☐ The program prints **true**.
- ☐ The program prints **false**.
- ☐ The program prints `null`.
- ☐ The program does not compile.
- ☐ None of the above.

3.4 (4 points) What happens when the following code is executed in Java? (Select one.)

```
String[] arr = { "Java", "OCaml"};  
String last = arr[arr.length];  
System.out.println(last);
```

- ☐ The program prints "OCaml".
- ☐ The program prints 2.
- ☐ The program prints null.
- ☐ An `ArrayIndexOutOfBoundsException` is thrown.
- ☐ None of the above.

3.5 (4 points) Which of the following is a valid way of creating a Java array with values "1", "2", and "3"? (Select all that apply, or "None of the above".)

- ☐ `String[] arr = {"1", "2", "3"};`
- ☐ `String arr = new String(new char[]{'1', '2', '3'});`
- ☐ `String[] arr = new int[]{1, 2, 3};`
- ☐ `String[] arr = { new String("1"), new String("2"), new String("3")};`
- ☐ None of the above.

4. Programming with Java Arrays (24 points total)

Recall that in Java, two dimensional arrays are represented as arrays of arrays. This means that in a 2D array of type `int[][]` there is no guarantee that the inner arrays are all the same length, or even nonnull.

For this problem, you will write a static function, called `extractRectangle` that copies a rectangular section starting from the upper-left corner of some input array.

```
/* Copy a rectangular array of size h * w from the input, starting at
   the upper left corner. Returns null if any part of the input is null,
   h or w is negative, or if the input array does not contain enough
   values. */
public static int[][] extractRectangle(int h, int w, int[][] input);
```

For example, the following test case demonstrates what happens when a rectangle can be extracted.

```
@Test
public void testExtractRectangle_validInput() {
    int[][] input = {
        {1, 2, 3},
        {4},
        {7, 8, 9, 10}
    };
    int[][] expected = {
        {1}, // truncate first row
        {4}  // copy entire second row
            // omit third row
    };
    assertEquals(expected, Exam.extractRectangle(2, 1, input));
}
```

And this test case demonstrates that the copy needs to start at the upper left corner. The method returns `null` because there are not enough values in the first row of the array.

```
@Test
public void testExtractRectangle_tooSmall() {
    int[][] input = {
        {1, 2}, // need to have 3 values, but only 2 present
        {4, 5, 6},
        {7, 8, 9}
    };
    assertNull(Exam.extractRectangle(2, 3, input));
}
```

- 4.1 (4 points) Fill in the expected result when a zero-by-zero rectangle is requested. (This should be a 2D array of size 0×0 .)

```
@Test
public void testRectangle0() {
    int[][] input = {
        {1, 2},
        {4, 5, 6},
        {7, 8, 9, 10}
    };
    int[][] expected =
        

    assertEquals(expected, Main.extractRectangle(0, 0, input));
}
```

- 4.2 (4 points) Complete a test case that demonstrates when the function should return **null** when requesting a zero-by-zero rectangle.

```
@Test
public void testRectangle1() {

    int[][] input =
        

    ;

    assertNull(Main.extractRectangle(0, 0, input));
}
```


- 4.3 (16 points) Now implement the method. Your method should never throw an `ArrayIndexOutOfBoundsException` exception or a `NullPointerException`. If either of these would happen, the method should return `null` instead.

*/** Remove a rectangular array of size h * w from the input, starting at the upper left corner. Returns null if any part of the input is null, h or w is negative, or if the input array does not contain enough values to remove. */*

```
public static int[][] removeRectangle(int h, int w, int[][] input) {
```

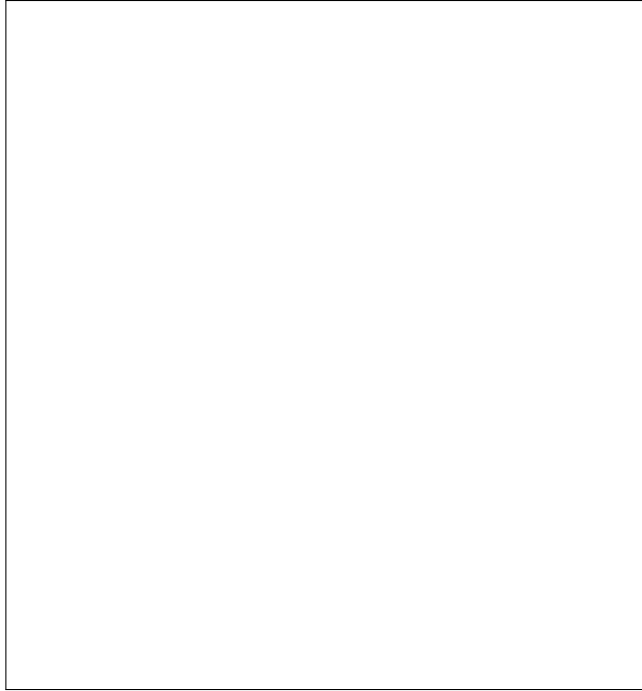
```
    if (  ) {  
        return null;  
    }
```

```
    int[][] output =  ;
```

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

```
        int start = 0;  
        if (i < h) {  
            start = w;  
        }
```

```
        if (  ) {  
            return null;  
        }
```



```
    }  
    return output;  
}
```

Scratch Space

*Use this page for work that you do not want us to grade. If you run out of space elsewhere in the exam and you **do** want to put something here that we should grade, make sure to put a clear note in the normal answer space for the problem in question.*

A Appendix: GUI Gctx interface (excerpt)

```
(** A widget-relative position *)
type position = int * int

(** A width and height paired together. *)
type dimension = int * int

(** Display text at the given position *)
val draw_string : gctx -> position -> string -> unit

(** Draw a line between the two specified positions *)
val draw_line : gctx -> position -> position -> unit

(** Draw an ellipse, centered at position with given x and y radii. *)
val draw_ellipse : gctx -> position -> int -> int -> unit
```

B Appendix: GUI Widget Interface (excerpt)

```
(* Widget *)

type widget = {
  repaint : Gctx.gctx -> unit;
  handle : Gctx.gctx -> Gctx.event -> unit;
  size : unit -> Gctx.dimension;
}

(** A horizontal group of widgets *)
val hlist : widget list -> widget

(*****)
(** {1 Label Widgets } *)
(*****)

(** A record of functions that allows us to read and write the string
associated with a label. *)
type label_controller = { get_label : unit -> string;
  set_label : string -> unit }

(*****)
(** {1 Event Listeners } *)
(*****)

(** An event listener processes events as they "flow" through
the widget hierarchy. *)
type event_listener = Gctx.gctx -> Gctx.event -> unit

(** Performs an action upon receiving a mouse click. *)
val mouseclick_listener : (unit -> unit) -> event_listener
```



```

(*****)
(** {2 Notifier } *)
(*****)

(** A notifier_controller is associated with a notifier widget.
It allows the program to add event listeners to the notifier. *)

type notifier_controller = { add_event_listener : event_listener -> unit; }

val notifier : widget -> widget * notifier_controller

(*****)
(** {3 Canvas } *)
(*****)

(** A widget that allows drawing to a graphics context *)

(** Canvases are just widgets with an associated notifier_controller.
The repaint method of a canvas is a parameter of the widget
constructor. *)
val bare_canvas : Gctx.dimension -> (Gctx.gctx -> unit) -> widget
val canvas      : Gctx.dimension -> (Gctx.gctx -> unit) ->
                                widget * notifier_controller

(*****)
(** {4 Value controller } *)
(*****)

(** A controller for a value associated with a widget.

This controller can read and write the value. It also allows
change listeners to be registered by the application. These listeners are
run whenever this value is set. *)
type 'a value_controller = {
  add_change_listener : ('a -> unit) -> unit;
  get_value           : unit -> 'a;
  change_value        : 'a -> unit
}

(** A utility function for creating a value_controller. *)
val make_controller : 'a -> 'a value_controller

```

C Appendix: Notifier widget implementation

This code uses the following function from the `List` library.

```
val iter : ('a -> unit) -> 'a list -> unit
```

```
1  (** A notifier widget is a widget "wrapper" that doesn't take up any
2     extra screen space -- it extends an existing widget with the
3     ability to react to events. It maintains a list of of
4     event_listeners that eavesdrop on the events propagated through the
5     notifier widget.
6     When an event comes in to the notifier, it is passed to each
7     event_listener in turn, and then passed to the child widget. *)
8  let notifier (w: widget) : widget * notifier_controller =
9    let listeners = {contents = []} in {
10      repaint = w.repaint;
11      handle =
12        (fun (g: Gctx.gctx) (e: Gctx.event) ->
13          List.iter (fun h -> h g e) listeners.contents;
14          w.handle g e);
15      size = w.size
16    }, {
17      add_event_listener =
18        fun (newl: event_listener) ->
19          listeners.contents <- newl :: listeners.contents
20    }
```

D Appendix: Java String class documentation

Constructors

<code>String(String original)</code>	Initializes a newly created <code>String</code> object so that it represents the same sequence of characters as the argument; in other words, the newly created string is a copy of the argument string.
<code>String(char[] value)</code>	Allocates a new <code>String</code> so that it represents the sequence of characters currently contained in the character array argument.

Methods

<code>int length()</code>	Returns the length of this string.
<code>char[] toCharArray()</code>	Converts this string to a new character array.
<code>boolean equals(Object obj)</code>	Compares this string to the specified object. The result is true if and only if the argument is not null and is a <code>String</code> object that represents the same sequence of characters as this object.