Frage **1**

Unvollständig

Erreichbare Punkte: 7,00

Hinweis:

Roter Text kennzeichnet Ihre Eingaben in der Konsole, Ihre Eingaben werden nicht tatsächlich rot sein.
Weißer Text kennzeichnet die Ausgaben Ihres Programms.

Aufgabe

Schreiben Sie ein Programm das es dem Benutzer ermöglicht in einer Priorityqueue Einträge zu verwalten. Ein Eintrag besteht aus eine Priorität und einer Nachricht. Die Nachricht darf bis zu 31 Zeichen lang sein und es sollen maximal 10 Einträge gespeichert werden können. Wird ein neuer Eintrag hinzugefügt, so überspringt er alle Einträge mit niedriger Priorität und wir hinter den Einträgen mit gleicher Priorität angereiht. Es gibt die folgenden 5 Prioritäten:

- HIGHEST (H)
- HIGH (h)
- NORMAL (n)
- LOW (l)
- LOWEST (L)

Ihr Programm soll ein Menü ausgeben, um folgende Aktionen beliebig oft durchzuführen:

1. einen neuen Eintrag hinzufügen (**enqueue**)
2. den nächsten Eintrag entnehmen und ausgeben (**dequeue**)
3. den Inhalt der Pirorityqueue ausgeben (**print**)
4. das Programm beenden (**exit**)

Beispiel Aus- und Eingaben:

Choose action: print queue (p), enqueue entry (e), dequeue entry (d) or exit (x): p

empty queue
Choose action: print queue (p), enqueue entry (e), dequeue entry (d) or exit (x): e

Choose priority: lowest (L), low (l), normal (n), high (h), highest (H): n

Choose message: First_Task

priority queue now contains 1 entry
Choose action: print queue (p), enqueue entry (e), dequeue entry (d) or exit (x): e

Choose priority: lowest (L), low (l), normal (n), high (h), highest (H): n

Choose message: Another_Task

priority queue now contains 2 entries
Choose action: print queue (p), enqueue entry (e), dequeue entry (d) or exit (x): p

n: First_Task
n: Another_Task
Choose action: print queue (p), enqueue entry (e), dequeue entry (d) or exit (x): e

Choose priority: lowest (L), low (l), normal (n), high (h), highest (H): h

Choose message: Urgent_Task

priority queue now contains 3 entries
Choose action: print queue (p), enqueue entry (e), dequeue entry (d) or exit (x): p

h: Urgent_Task
n: First_Task
n: Another_Task
Choose action: print queue (p), enqueue entry (e), dequeue entry (d) or exit (x): d

Message: Urgent_Task
priority queue now contains 2 entries
Choose action: print queue (p), enqueue entry (e), dequeue entry (d) or exit (x): d

Message: First_Task
priority queue now contains 1 entry
Choose action: print queue (p), enqueue entry (e), dequeue entry (d) or exit (x): x

Allgemein zur Formatierung:
In CodeRunner wird jede einzelne Eingabe von einem Zeilenumbruch bestätigt, der in der Ausgabe aber nicht sichtbar ist. Wenn Sie außerhalb von CodeRunner das Programm starten, selber Werte eingeben (siehe Beispielausgabe) und dabei Zeilenumbrüche produzieren, wirkt sich das auf die Ausgabe aus (mehr Zeilenumbrüche als beim automatischen Testen in CodeRunner).

Allgemein zu Ausgaben:
Geben Sie Zeilenumbrüche immer vor einer neue Zeile aus und nicht am Ende. Wenn Sie sich daran halten, sollte die Formatierung leichter zur Übereinstimmung mit der erwarteten Ausgabe gebracht werden können. Das Programm startet daher auch mit einem Zeilenumbruch.

Lesen Sie die gesamte Angabe bevor Sie mit der Implementierung starten.

Folgende Bedingungen **muss** ihr Programm erfüllen:

- Verwenden Sie eine Enumeration (`priority`) zur Definition der Prioritäten:
  - `highest`
  - `high`
  - `normal`
  - `low`

- `lowest`

- Verwenden Sie eine Struktur um einen Eintrag (`entry`) zu speichern:
  - Enumeration Priorität (`priority`)
  - Char-Array Nachricht (`message`)
- Verwenden Sie eine weitere Struktur um die Priorityqueue (`pqueue`) zu speichern:
  - Array von Einträgen (`entries`) für maximal 10 Einträge.
  - Optional können Sie hier noch weitere Informationen speichern, wenn Sie dies für nötig halten.

Schreiben Sie folgende Funktionen zur Interaktion mit Priorityqueues:

- `int isEmpty(struct pqueue* pqueue)`: Die Funktion gibt zurück, ob die übergebene Priorityqueue leer ist. Rückgabewert `1` (oder beliebige Zahl ungleich 0) wenn keine Einträge vorhanden sind, ansonsten wird `0` zurück gegeben.

- `int isFull(struct pqueue* pqueue)`: Die Funktion gibt zurück, ob die übergebene Priorityqueue voll ist. Rückgabewert `1` (oder beliebige Zahl ungleich 0) wenn bereits 10 Einträge vorhanden sind, ansonsten wird `0` zurück gegeben.

- `void enqueue(struct pqueue* pqueue, struct entry entry)`: Die Funktion fügt einen neuen Eintrag `entry` in die übergebene Priorityqueue `pqueue` ein. Der Eintrag wird an jene Stelle eingefügt, **dass alle Einträge mit gleicher oder höherer Priorität vor dem neuen Eintrag liegen.** Alle dahinter liegenden Einträge (mit niedrigerer Priorität) müssen um eine Stelle verschoben werden um Platz zu schaffen.
  Ist die Priorityqueue vor dem Einfügen bereits voll, so wird der Eintrag nicht eingefügt und folgende Fehlermeldung ausgegeben:
  priority queue already full!
  Nach erfolgreichem Einfügen des Eintrags soll die Anzahl der Einträge in der Priorityqueue ausgegeben werden.
  priority queue now contains 1 entry

- `void dequeue(struct pqueue* pqueue)`: Die Funktion entfernt das erste Element aus der Piorityqueue und gibt dessen Nachricht aus. Alle weiteren Elemente werden um eine Position nach vorne verschoben.
  Die Ausgabe soll wie folgt formatiert sein:
  Message: <entry.message>
  Ist die Priorityqueue leer wird folgende Fehlermeldung ausgegeben:
  priority queue is empty!
  Nach erfolgreichem Entfernen eines Eintrages soll auch wieder die Anzahl der Einträge in der Priorityqueue ausgegeben werden.
  priority queue now contains 0 entries

Schreiben Sie folgende Funktionen zur Ausgabe:

- `void printQueue(struct pqueue* pqueue)`: Die Funktion gibt die als Parameter übergebene Priorityqueue aus. Dabei soll jeder Eintrag in einer neuen Zeile im Folgenden Format ausgegeben werden: "<Priorität>: <Nachricht>". Die Einträge sollen in der selben Reihenfolge ausgegeben werden, in der sie in der Priorityqueue gespeichert sind.
  Beispiel:
  H: Message_with_highest_priority.
  h: Message_with_high_priority.
  n: Message_with_normal_priority.
  n: Another_message_with_normal_priority.
  l: Message_with_low_priority.
  Sind keine Einträge vorhanden soll folgender Text ausgegeben werden:
  empty queue

Allgemein zu Eingaben:
Geben Sie vor jeder Eingabe noch ein Leerzeichen mit aus, um die Eingabe visuell zu trennen.
Eingegebene Zeilenumbrüche und führende Leerzeichen sollen ignoriert werden. Wenn etwas Ungültiges eingegeben wurde, soll "Input invalid! Try again: " ausgegeben und erneut eingelesen werden. Sie können sich aber darauf verlassen, dass Zeichen eingegeben werden, wenn Zeichen erwartet werden und Text eingegeben wird, wenn Text erwartet wird.
Weiters dürfen Sie davon ausgehen, dass nie mehr als die maximalen 31 Zeichen an Text eingegeben werden.

Schreiben Sie folgende Funktionen zur Eingabe:

- `char getMenu()`: Die Funktion erfragt welche Aktion ausgeführt werden soll und liefert ein entsprechendes Zeichen zurück. Die Eingabe soll so lange wiederholt werden, bis ein gültiges Zeichen eingegeben wurde.

  Choose action: print queue (p), enqueue entry (e), dequeue entry (d) or exit (x): w

  Input invalid! Try again: x
- `struct entry getNewEntry()`: Die Funktion erfragt die Priorität und Nachricht und liefert einen Eintrag mit den entsprechenden Werten

zurück.

Choose action: print queue (p), enqueue entry (e), dequeue entry (d) or exit (x): e

Choose priority: lowest (L), low (l), normal (n), high (h), highest (H): a

Input invalid! Try again: L

Choose message: Do_something!

Die Nachricht besteht immer aus einem zusammenhängenden String ohne whitespaces (also auch keine Leerzeichen). Dies ermöglich die Verwendung von scanf zum Einlesen. (Wenn Sie Leerzeichen unterstützen wollen, können sie fgets verwenden, müssen aber manuell zuerst whitespaces vom input und den Zeilenumbruch aus dem String entfernen. Dies ist für den Test explizit nicht gefordert).

Alle Funktionsdefinitionen (Funktionsname, Parameteranzahl, Typen der Parameter) müssen der Angabe entsprechen.
Parameternamen die in der Angabe in Klammern angegeben sind, sind Vorschläge und können verändert werden.

Das Verwenden von globalen Variablen ist nicht erlaubt!

Eine **Main Funktion** soll eine Priorityqueue initialisieren und die implementierten Funktionen aufrufen um die gewünschte Funktionalität des Programmes sicherzustellen.

Hinweise

- Setzen Sie die Angabe um und implementieren Sie nicht anhand der Testfälle.
- Nutzen Sie eine lokale Entwicklungsumgebung und testen Sie Ihr Programm selbst, bevor Sie CodeRunner nutzen.
- Es empfiehlt sich, zuerst die Enumeration und die Strukturen zu definieren, dann die Funktione zur Ausgabe der Priorityqueue zu implementieren, da Sie diese zum Debuggen verwenden können.
- Nutzen Sie bereits implementierte Funktionen und vermeiden Sie redundanten Code.
- Weisen Sie den Prioritäten im Enum Werte zu, um Prioritäten vergleichen zu können.
- Weiters empfiehlt es sich für das Mapping von Buchstaben- auf Integer-Prioritäten eigene Funktionen zu impementieren.

Antwort: (Abzugssystem: 0 %)

```
1
```

Prüfen

◄ Sammlung: Fragen zum Eigenstudium O

Direkt zu:

Peer Review Zwischentest 2 ►