Outline

fig12_13.c (Part 1 of 7)

```c
/* Fig. 12.13: fig12_13.c
   Operating and maintaining a queue */

#include <stdio.h>
#include <stdlib.h>

/* self-referential structure */
struct queueNode {
   char data;                   /* define data as a char */
   struct queueNode *nextPtr; /* queueNode pointer */
}; /* end structure queueNode */

typedef struct queueNode QueueNode;
typedef QueueNode *QueueNodePtr;

/* function prototypes */
void printQueue( QueueNodePtr currentPtr );
int isEmpty( QueueNodePtr headPtr );
char dequeue( QueueNodePtr *headPtr, QueueNodePtr *tailPtr );
void enqueue( QueueNodePtr *headPtr, QueueNodePtr *tailPtr,
              char value );
void instructions( void );

/* function main begins program execution */
int main()
{
```

Outline

fig12_13.c (Part 2 of 7)

```c
   QueueNodePtr headPtr = NULL; /* initialize headPtr */
   QueueNodePtr tailPtr = NULL; /* initialize tailPtr */
   int choice;                  /* user's menu choice */
   char item;                   /* char input by user */

   instructions(); /* display the menu */
   printf( "? " );
   scanf( "%d", &choice );

   /* while user does not enter 3 */
   while ( choice != 3 ) {

      switch( choice ) {

         /* enqueue value */
         case 1:
            printf( "Enter a character: " );
            scanf( "\n%c", &item );
            enqueue( &headPtr, &tailPtr, item );
            printQueue( headPtr );
            break;

         /* dequeue value */
         case 2:

```

Outline

fig12_13.c (Part 3 of 7)

```c
            /* if queue is not empty */
            if ( !isEmpty( headPtr ) ) {
               item = dequeue( &headPtr, &tailPtr );
               printf( "%c has been dequeued.\n", item );
            } /* end if */

            printQueue( headPtr );
            break;

         default:
            printf( "Invalid choice.\n\n" );
            instructions();
            break;

      } /* end switch */

      printf( "? " );
      scanf( "%d", &choice );
   } /* end while */

   printf( "End of run.\n" );

   return 0; /* indicates successful termination */

} /* end main */

```

Outline

fig12_13.c (Part 4 of 7)

```c
/* display program instructions to user */
void instructions( void )
{
   printf ( "Enter your choice:\n"
           "   1 to add an item to the queue\n"
           "   2 to remove an item from the queue\n"
           "   3 to end\n" );
} /* end function instructions */

/* insert a node a queue tail */
void enqueue( QueueNodePtr *headPtr, QueueNodePtr *tailPtr,
              char value )
{
   QueueNodePtr newPtr; /* pointer to new node */

   newPtr = malloc( sizeof( QueueNode ) );

   if ( newPtr != NULL ) { /* is space available */
      newPtr->data = value;
      newPtr->nextPtr = NULL;

      /* if empty, insert node at head */
      if ( isEmpty( *headPtr ) ) {
         *headPtr = newPtr;
      } /* end if */
```

Outline

fig12_13.c (Part 5 of 7)

```c
      else {
         ( *tailPtr )->nextPtr = newPtr;
      } /* end else */

      *tailPtr = newPtr;
   } /* end if */
   else {
      printf( "%c not inserted. No memory available.\n", value );
   } /* end else */

} /* end function enqueue */

/* remove node from queue head */
char dequeue( QueueNodePtr *headPtr, QueueNodePtr *tailPtr )
{
   char value;           /* node value */
   QueueNodePtr tempPtr; /* temporary node pointer */

   value = ( *headPtr )->data;
   tempPtr = *headPtr;
   *headPtr = ( *headPtr )->nextPtr;

   /* if queue is empty */
   if ( *headPtr == NULL ) {
      *tailPtr = NULL;
   } /* end if */

   free( tempPtr );

   return value;

} /* end function dequeue */

/* Return 1 if the list is empty, 0 otherwise */
int isEmpty( QueueNodePtr headPtr )
{
   return headPtr == NULL;

} /* end function isEmpty */

/* Print the queue */
void printQueue( QueueNodePtr currentPtr )
{

   /* if queue is empty */
   if ( currentPtr == NULL ) {
      printf( "Queue is empty.\n\n" );
   } /* end if */
   else {
      printf( "The queue is:\n" );

```

Outline

fig12_13.c (Part 6 of 7)

Outline

fig12_13.c (Part 7 of 7)

```c
      /* while not end of queue */
      while ( currentPtr != NULL ) {
         printf( "%c --> ", currentPtr->data );
         currentPtr = currentPtr->nextPtr;
      } /* end while */

      printf( "NULL\n\n" );
   } /* end else */

} /* end function printQueue */
```

Program Output (Part 1 of 2)

```
Enter your choice:
   1 to add an item to the queue
   2 to remove an item from the queue
   3 to end
? 1
Enter a character: A
The queue is:
A --> NULL

? 1
Enter a character: B
The queue is:
A --> B --> NULL

? 1
Enter a character: C
The queue is:
A --> B --> C --> NULL
```

Outline

Program Output (Part 2 of 2)

```
? 2
A has been dequeued.
The queue is:
B --> C --> NULL

? 2
B has been dequeued.
The queue is:
C --> NULL

? 2
C has been dequeued.
Queue is empty.

? 2
Queue is empty.

? 4
Invalid choice.

Enter your choice:
   1 to add an item to the queue
   2 to remove an item from the queue
   3 to end
? 3
End of run.
```