University of Nevada, Las Vegas Computer Science 477/677 Fall 2023

Answers to Assignment 2: Due Thursday September 14 2023

1. Fill in the Blanks

   (a) The operators of type stack are _____, _____, and _____.

   (b) The operators of type array are _____ and _____.

   (c) Three kinds of priority queues are **stack**, **queue**, and **heap**.

   There may be others, but any answer different from these must be really good. Also "maxheap" and "minheap" do not count as two separate answers.

2. Implementations of Stack

   (a) Write a C++ program that contains an array implementation of stack of integers. You may use either a struct or a class. Your stack should have a capacity of 10 integers. For ease of grading, do not submit a handwritten solution.

```
const int N=10;

struct stack
 {
  int item[N];
  int size = 0;
 };

bool empty(stack s)
 {
  return s.size == 0;
 }

void push(stack&s, int newitem)
 {
  s.item[s.size] = newitem;
  s.size++;
 }

int pop(stack&s)
 {
  s.size--;
  return s.item[s.size];
 }
```

```
int main()
 {
  stack s;
  push(s,5);
  push(s,7);
  push(s,-3);
  cout << pop(s) << endl;
  push(s,9);
  while(not empty(s))
   cout << pop(s) << " ";
  cout << endl;
  return 1;
 }
```

(b) Write a C++ program that contains a linked list array implementation of stack of integers. You may use either a struct or a class. For ease of grading, do not submit a handwritten solution.

```
struct stacknode
 {
  int item;
  stacknode*link;
 };

typedef stacknode*stack;

bool empty(stack s)
 {
  return s == NULL;
 }

void push(stack&s, int newitem)
 {
  stack temp = new stacknode;
  temp->item = newitem;
  temp->link = s;
  s = temp;
 }

int pop(stack&s)
 {
  assert(not empty(s));
  int rslt = s->item;
  s = s->link;
  return rslt;
 }
```

```
int main()
 {
  stack s = NULL;
  push(s,5);
  push(s,7);
  push(s,-3);
  cout << pop(s) << endl;
  push(s,9);
  while(not empty(s))
   cout << pop(s) << " ";
  return 1;
 }
```

For both implementations, the output should be:

```
-3
9 7 5
```

3. Implementations of Queue

   (a) Write a C++ program that contains an array implementation of queue of integers. You may use either a struct or a class. Your queue should hold 10 integers. For ease of grading, do not submit a handwritten solution.

```
const int N=10;

struct queue
 {
  int item[N];
  int front = 0;
  int rear = 0;
 };

bool empty(queue q)
 {
  return q.rear == q.front;
 }

void enqueue(queue&q, int newitem)
 {
  assert(q.rear <= N);
  q.item[q.rear] = newitem;
  q.rear++;
 }
```

```
int dequeue(queue&q)
 {
  assert(not empty(q));
  int rslt = q.item[q.front];
  q.front++;
  return rslt;
 }

int main()
 {
  queue q;
  enqueue(q,5);
  enqueue(q,7);
  enqueue(q,-3);
  cout << dequeue(q) << endl;
  enqueue(q,9);
  while(not empty(q))
   cout << dequeue(q) << " ";
  cout << endl;
  return 1;
 }
```

(b) Write a C++ program that contains a linked list implementation of queue of integers. You may use either a struct or a class.

For both implementations, start with an empty queue, then execute the following, in this order, where q is a variable of type queue. For ease of grading, do not submit a handwritten solution.

```
    // Circular Linked List with Dummy Node Implementation

struct circlist
 {
  int item;
  circlist*link;
  circlist()
   {
    link = this;
   }
 };

typedef circlist* queue;
```

```
bool empty(queue q)
 {
  return q == q->link;
 }

void enqueue(queue&q, int newitem)
 {
  queue temp = new circlist;
  temp->link = q->link;
  q->link = temp;
  q->item = newitem;
  q = temp;
 }

int dequeue(queue&q)
 {
  assert(not empty(q));
  int rslt = q->link->item;
  q->link = q->link->link;
  return rslt;
 }

int main()
 {
  queue q = new circlist;
  enqueue(q,5);
  enqueue(q,7);
  enqueue(q,-3);
  cout << dequeue(q) << endl;
  enqueue(q,9);
  while(not empty(q))
   cout << dequeue(q) << " ";
  cout << endl;
  return 1;
 }
```

For both implementations, the output should be:

```
5
7 -3 9
```

4. There is a stack algorithm for converting an infix expression to postfix. (You can find it if you look.) Write that algorithm in pseudocode. Use the algorithm to convert the infix expression $x * (-y - z) + x$ to postfix, showing the stack at each step. Hint: in postfix or prefix expressions, subtraction is indicated with a minus sign, but negation by a tilde, such as $\sim x$ (prefix) or $x \sim$ (postfix). (This algorithm is used every day by compilers.) You may write your answer by hand if you wish.

   The algorithm is described in the handout. Each step of the computation given below is determined by one entry of the matrix in that handout.

   The following matrix shows one row for each step of the algorithm. Each row consists of the stack, with the top on the right, followed by the input file written from left to right, followed by the output file.

Stack	Input	Output
$\$$	$x * (\sim y - z) + x \$$	
$\$$	$*(\sim y - z) + x \$$	x
$\$*$	$(\sim y - z) + x \$$	x
$\$ * ($	$\sim y - z) + x \$$	x
$\$ * (\sim$	$y - z) + x \$$	x
$\$ * (\sim$	$-z) + x \$$	$x\, y$
$\$ * ($	$-z) + x \$$	$x\, y \sim$
$\$ * (-$	$z) + x \$$	$x\, y \sim$
$\$ * (-$	$) + x \$$	$x\, y \sim z$
$\$ * ($	$) + x \$$	$x\, y \sim z -$
$\$ *$	$+ x \$$	$x\, y \sim z -$
$\$$	$+ x \$$	$x\, y \sim z - *$
$\$ +$	$x \$$	$x\, y \sim z - *$
$\$ +$	$\$$	$x\, y \sim z - *\, x$
$\$$	$\$$	$x\, y \sim z - *\, x +$