University of Nevada, Las Vegas Computer Science 477/677 Fall 2025
Answers to Assignment 2: Due Saturday September 13, 2025

Follow our TA Louis DuMontet's (`dumontet@unlv.nevada.edu`) instructions on how to turn in the assignment.

1. Fill in the blanks.

   (a) No good programmer would ever use an unsorted list as a search structure. **F** (True or False.)

   (b) Heapsort is a fast version of **selection sort.**

   (c) Treesort is a fast version of **insertion sort.**

   (d) Shell sort is inspired by **bubblesort** but is faster.

   (e) The items of a priority queue represent **unfulfilled obligations.**

2. State the asysmptotic time complexity, in terms of n, of each of these code fragments. Use Θ notation.

   (a)
   ```
   for(int i = 0; i < n; i++)
    for(int j = 0; j < n; j++)
     cout << "Hello world." << endl;;
   ```
   $\Theta(n^2)$

   (b)
   ```
   for(int i = n; i > 0; i = i/2)
    cout << "Hello world." << endl;;
   ```
   $\Theta(\log n)$

   (c)
   ```
   for(int i = 1; i < n; i++)
    for(int j = 1; j < n; j = 2*j)
     cout << "Hello world." << endl;;
   ```
   $\Theta(n \log n)$

   (d)
   ```
   for(int i = 1; i < n; i = 2*i)
    for(int j = 0; j < i; j++)
     cout << "Hello world." << endl;;
   ```
   Warning: this one is tricky.

   $\Theta(n)$

3. The following is C++ code for which algorithm?

```
void sort(int x[n])
 {
  for(int i = 0; i+1 < n; i++)
   for(int j = i+1; j < n; j++)
    if(x[j] < x[i]) swap(x[i],x[j]);
 }
```

selection sort.

4. The following is C++ code for which algorithm?

```
void sort(int x[n])
 {
  for(int i = 1; i < n; i++)
   for(int j = i; j > 0 and x[j] < x[j-1]; j--)
    swap(x[j],x[j-1]);
 }
```

insertion sort.

5. The following is C++ code for which algorithm?

```
void sort(int x[n])
 {
  for(int j = n-1; j > 0; j--)
   for(int i = 0; i < j; i++)
    if(x[i+1] < x[i]) swap(x[i],x[i+1]);
 }
```

bubblesort.

6. The following is C++ code for which algorithm?

```
bool find(int sought, int x[n])
 // input condition: x is in increasing order
 {
  int lo = 0;
  int hi = n;
  while(lo+1 < hi)
   {
    int mid = (lo+hi)/2;
    if(sought < x[mid]) hi = mid;
    else lo = mid;
   }
  return x[lo] == sought;
 }
```

binary search.

7. Name two different divide-and-conquer sorting algorithms.

   mergesort
   quicksort

8. It is not clear how to write a loop invariant for a `for` loop, since the loop index is undefined outside the body of the loop. We need to first replace all `for` loops by while or `loop ...until` loops. Here is an modified version of Problem 3. Write the loop invariant for each of its two loops.

```
void sort(int x[n])
 {
  int i = 0;
  while(i+1 < n)
   {
    int j = i+1;
    while(j < n)
     {
      if(x[j] < x[i]) swap(x[i],x[j]);
      j++;
     }
    i++;
   }
 }
```

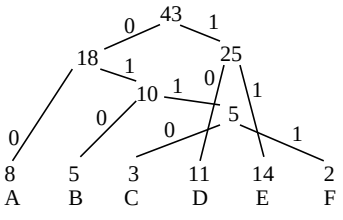
9. The C++ code below implements a function. What does that function compute?

```
int gcd(int n, int m)
 {
  if(n < 0) return gcd(-n,m);
  else if(m < 0) return gcd(n,-m);
  else if(n < m) return gcd(m,n);
  else if(m > 0) return gcd(m,n%m);
  else return n;
 }
```

   The function computes the greatest common divisor of two integers which are not both zero. You learned about it in the fifth grade. To reduce a fraction to its lowest terms, divide both numerator and denominator by their greatest common divisor. For example, gcd(12,18) = 6, hence $\frac{12}{18} = \frac{2}{3}$.

10. Find an optimal binary prefix-free code on the alphabet A,B,C,D,E,F where frequencies are given in the following table.

A	8	00
B	5	010
C	3	0110
D	11	10
E	14	11
F	2	0111

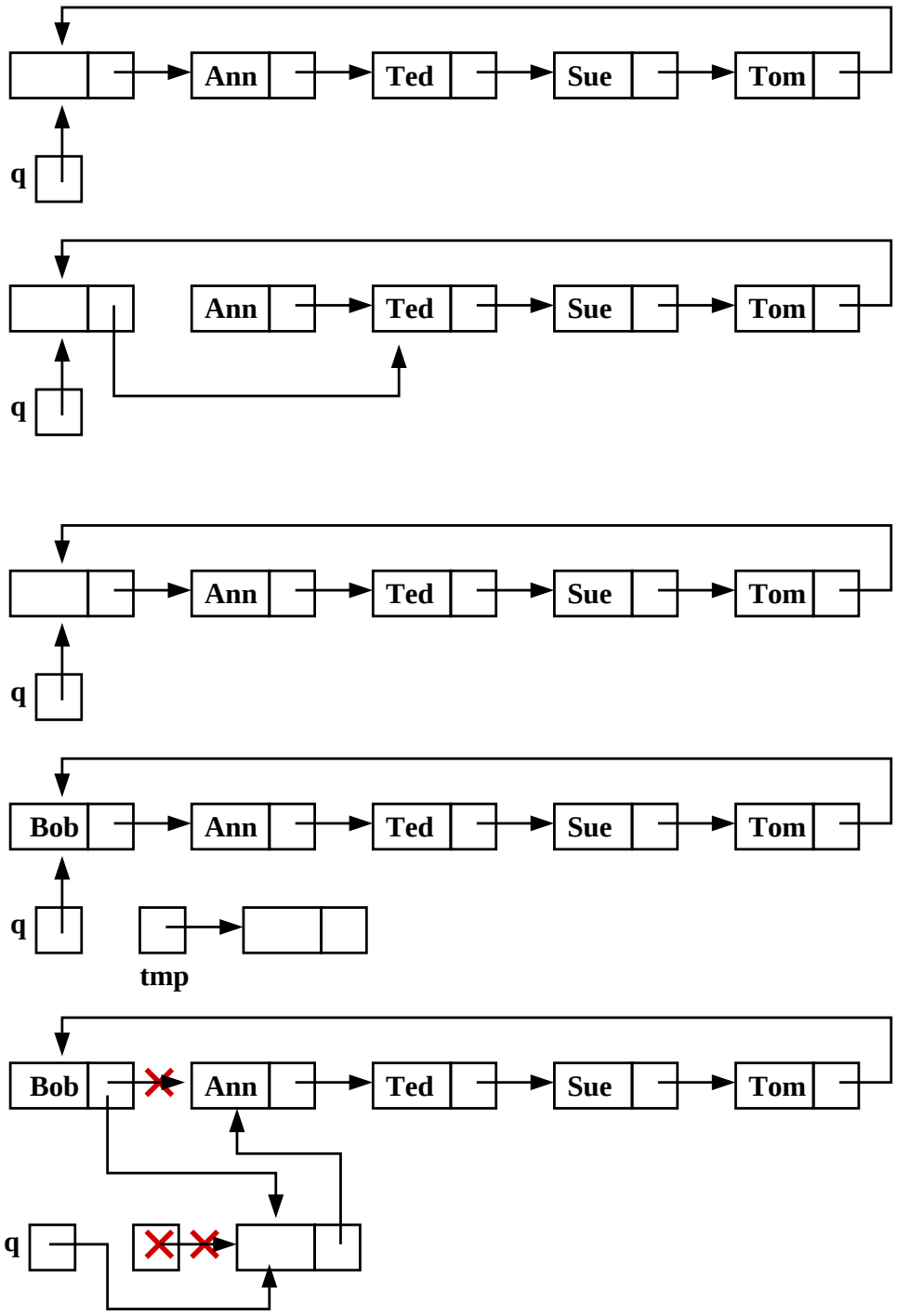


11. Illustrate a queue implemented as a circular linked list with a dummy node. The contents of the queue, from front to rear, should be the four items Ann, Ted, Sue, Tom.

    (a) Show the steps of dequeue, using the same example.

    (b) Redraw the original figure, and show the steps of enqueue where the item Bob is added to the queue.

    In each case, do not erase deleted objects. Instead, cross them out.



The following is a partial array implementation of the ADT stack of integer. Fill in the missing code for empty, push, and pop.

```
struct stak
 {
  int item[N];
  int size;
 };

void init(stak&S)
 {
  S.size = 0;
 }

bool empty(stak S)
 {
  return S.size == 0;
 }

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

int pop(stak&S)
 {
  assert(not empty(S));
  return S.item[--S.size];
 }
```