IS2610 Data Structure Hw3. Solution
By Kai-Hong Liu

1.
Inorder CBDAGFHEI
Preorder ABCDEFGHI
Postorder CDBGHFIEA

2.
2.a
Recursive:
Preorder

```
void traverse(link h, void (*visit)(link))
  {
    if (h == NULL) return;
    (*visit)(h);
    traverse(h->l, visit);
    traverse(h->r, visit);
  }
```

Postorder

```
void traverse(link h, void (*visit)(link))
  {
    if (h == NULL) return;
    traverse(h->l, visit);
    traverse(h->r, visit);
    (*visit)(h);

  }
```

Inorder

```
void traverse(link h, void (*visit)(link))
  {
    if (h == NULL) return;
    traverse(h->l, visit);
    (*visit)(h);
    traverse(h->r, visit);

  }
```

Non-Recursive
Preorder

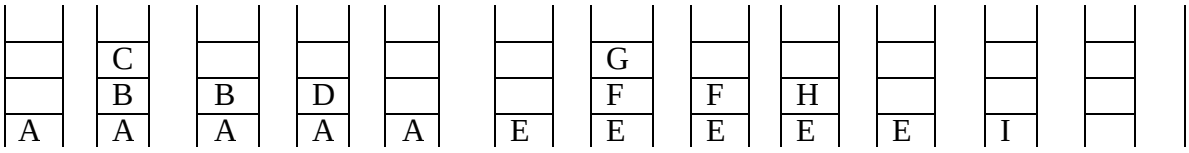
```
void traverse(link h, void (*visit)(link))
  {
    STACKinit(max); STACKpush(h);
    while (!STACKempty())
      {
        (*visit)(h = STACKpop());
        if (h->r != NULL) STACKpush(h->r);
        if (h->l != NULL) STACKpush(h->l);
      }
  }
```

```
Inorder
void traverse(link h, void (*visit)(link))
  {
    STACKinit(max); STACKpush(h);
    while (!STACKempty())
      {
        while (h-l!=NULL)
        {       while (h->l != NULL) & (h is not visited)
                {
                STACKpush(h->l);
                h=h->l;
                }

                (*visit)(h = STACKpop());

                 if (h->r != NULL) STACKpush(h->r)
                {
                STACKpush(h->r);
                h=h->r:
                }
        }

        (*visit)(h = STACKpop());
        if (h->r != NULL)
        {
                STACKpuch(h->r);
                h=h->r;
        }

      }
  }
```

```
Postorder
void traverse(link h, void (*visit)(link))
  void traverse(link h, void (*visit)(link))
  {
    STACKinit(max); STACKpush(h);
    while (!STACKempty())
      {
        while (h-l!=NULL)
        {       while (h->l != NULL) & (h->l is not visited)
                {
                STACKpush(h->l);
                h=h->l;
                }

                (*visit)(h = STACKpop());


                h = STACKpop();
                /*Here only popup the item get the pointer of the item
                but not visit it –don't print out the item yet. So we
                can get access the right child.*/

                STACKpuch(h);
                /* need to puch back that item to the stack so we can
                visit it later after we visit the right child*/

                if (h->r != NULL) & (h->r is not visited)

                {
                STACKpuch(h->r);
                h=h->r;
                }
        }
        if (h->r != NULL) & (h->r is not visited)

        {
                STACKpuch(h->r);
                h=h->r;
        }

        (*visit)(h = STACKpop());

      }
  }
```

2.b Inorder stack content

1.



Visited (print out) **C B D A G F H E I**

3.

```
#define key(A) (A)
#define less(A, B) (key(A) < key(B))
#define exch(A, B) { Item t = A; A = B; B = t; }

void selectionSort(Item a[], int L, int r)
{
  int i=L;
 if ( L == r-1) return;

  else{
         int min = L;
        for (i = i+1; i <= r; i++)
          {  if (less(a[i], a[min]) min = i;}

         exch(a[i], a[min]);
         selectionSort(Item a[], L+1, r);
         }

}
```

Iterative method for loop executes n – 1 times

- For each of n – 1 calls, inner loop executes n – 2 times
- (n – 1) + (n – 2) + …+ 1 = n(n – 1)/2 = O(n2)

Recursive selection sort performs same operations

- Also O(n2)

4. Selection Sort in general need n(n-1)/2 comparisons but only exchange n-1 times.
For bubble sort need n(n-1)/2 comparisons and in the worst case like the list is in reverse order will need n(n-1)/2 exchange. For the best case-the list is in sorted order not exchange needed..

		Comparison	Exchange
a. 12345678910	selection sort	10(10-1)/2 =45	10-1=9
	bubble sort	10(10-1)/2 =45	10(10-1)/2 =45

So selection faster.

b. 3333333333	selection sort	10(10-1)/2 =45	9
	bubble sort	10(10-1)/2 =45	0

the bubble sort is faster

c. 10987654321	selection sort	10(10-1)/2	=45	9
	bubble sort	10(10-1)/2	=45	0

the bubble sort is faster.