Assertion Insertion

1. Exercise 1: Binary tree building

```
typedef enum {false, true} Bool;
typedef struct node {
            int val;
            struct node *pndLeft;
            struct node *pndRight;
         } Node;


Node *PndNew(int val)
{
   Node *pndNew;

   *pndNew = (Node *) malloc(sizeof(Node));

   if (pndNew != NULL)
      {

         pndNew->pndLeft = pndNew->pndRight = NULL;

         pndNew->val = val;

      }

   return pndNew;
}



void InsertVal(int val, Node **ppnd)
{
   Node **ppndCur;

   ppndCur = ppnd;

   while (*ppndCur != NULL)
      {

         ppndCur = (val <= (*ppndCur)->val) ?
                     &((*ppndCur)->pndLeft) :
                     &((*ppndCur)->pndRight);

      }
```

```
   *ppndCur = PndNew(val);

   if (*ppndCur = NULL)
      {

         *ppnd = NULL;

      }

}



Bool FBuildBinTree(int rgval, int ivalMac, Node **ppnd)
{
   int ival;


   for (ival = 0; ival < ivalMac; ival++)
      {

         InsertVal(rgval[ival], ppnd);

         if (*ppnd == NULL)
            {

               return false;
            }

      }

   return true;
}
```

Hey! This code could potentially leak memory. Can you explain when and how?

2. Exercise 2: Finite state machine simultation

```
typedef enum {false, true} Bool;


Bool FInRgi(int i, int *rgi, int iMac)
{
   int iCur;

   for (iCur = 0; iCur < iMac; iCur++)
      {

         if (i == rgi[iCur])
            {

               return true;
            }

      }

   return false;
}



Bool FAcceptedByFsm(int *rgiInput, int iInputMac,
                    int *rgiTransit, int iTransitMac,
                    int *rgiAccept, int iAcceptMac)
{
   int iState;
   int *piInputCur, *piInputMac;

   iState = 0;

   piInputCur = rgiInput;

   piInputMac = rgiInput + iInputMac;

   while (piInputCur < piInputMac)
      {

         iState = rgiTransit[*piInputCur++];

      }

   return FInRgi(iState, rgiAccept, iAcceptMac);
}
```