HW12:

---

interface:

```
element* CreateMinHeap(int* top);
void Insert(element* target_heap,int* top);
int RemoveMinHeap(element* target_heap,int *top);
void ChangePrior(element* target_heap,int*top);
```

第一步：利用 CreateMinHeap 初始化 Heap 陣列

```
element* heap;
    int top;
    heap=CreateMinHeap(&top);
```

之後，便能操作另外三種函式：

```
void Insert(element* target_heap,int* top);
int RemoveMinHeap(element* target_heap,int *top);
void ChangePrior(element* target_heap,int*top);
```

這些函式被呼叫時，會同時要求使用者輸入欲插入的數值，因此使用上不用於程式碼

中宣告變數，並一再讀入,在使用上只需要知道欲插入多少數值即可以 for loop 撰

寫。

程式碼：CreateMinHeap

```
element* CreateMinHeap(int* top){
    static element minheap[MAX_SIZE];//must use static variable when
returning pointer
    *top=1;
    return &minheap[0];
};
```

說明：這個函式是用來初始化 heap ，將宣告的空間以指標回傳。

程式碼: Insert

```
 int i=*top;
   (target_heap+(i))->key=insert_key;
   int temp;
   for(;i>1;){
       if(i%2==1){//insert to the right child
           if((target_heap+i)->key<(target_heap+(i-1)/2)->key)
           {
               temp=(target_heap+i)->key;
               (target_heap+i)->key=(target_heap+(i-1)/2)->key;
               (target_heap+(i-1)/2)->key=temp;
               i=(i-1)/2;
           }else{break;}
      }else
       if(i%2==0){//insert to the left child
           if((target_heap+i)->key<(target_heap+i/2)->key)
           {
               temp=(target_heap+i)->key;
               (target_heap+i)->key=(target_heap+i/2)->key;
               (target_heap+i/2)->key=temp;
               i/=2;
           }else{break;}
       }
   }
   ++*top;
```

說明: 將新的數值放在 Heap array 的最上層之後，進行向上(bottom-up)的整

理，如果新的值小於 parent，則兩者交換，直到最頂端(前端)。

程式碼:RemoveMinHeap

```
int RemoveMinHeap(element* target_heap,int *top){

   int temp,min_result=(target_heap+1)->key;

   (target_heap+1)->key=(target_heap+*top-1)->key;
```

```
    for(int i=1;i<*top;){

        if((target_heap+i*2)->key==0){
            //left child==0( suggest right child ==0 )
            break;

        }else if((target_heap+i*2)->key!=0 &&(target_heap+i*2+1)-
>key==0){
            // left child !=0 and right child==0
            (target_heap+i)->key=(target_heap+i*2)->key;
            i=2*i;
        }
        else if((target_heap+i*2)->key!=0 &&(target_heap+i*2+1)-
>key!=0){
            //right and left both !=0

            if( (target_heap+i*2)->key<(target_heap+i*2+1)->key){
                //when left child is smaller
                if((target_heap+i*2)->key < (target_heap+i)->key){
                    //and the parent is larger than left
                    //swape parent and left child
                    temp=(target_heap+i)->key;
                    (target_heap+i)->key=(target_heap+i*2)->key;
                    (target_heap+i*2)->key=temp;
                    i=i*2;
                }//if left child is smaller and the parent is smaller
than left child,
                else break;//do nothing

            }else if( (target_heap+i*2)->key>= (target_heap+i*2+1)-
>key){
                //when right child is smaller
                if((target_heap+i*2+1)->key < (target_heap+i)->key){
                    //and the parent is largert than right
                    //swape parent and right child
                    temp=(target_heap+i)->key;
                    (target_heap+i)->key=(target_heap+i*2+1)->key;
                    (target_heap+i*2+1)->key=temp;
```

```
                    i=i*2+1;
                }else break;
            }
        }
    }
    --*top;
    (target_heap+*top)->key=0;
    return min_result;
};
```

說明: 先將陣列中最前端與最尾端兩個數值交換，之後進行向下(top-down)的方式與 left child, right child 進行比較，最小的會與原 parent 交換成為新 parent，並在交換後重複向下執行直到 array 最尾端。

程式碼: ChangePrior

```
void ChangePrior(element* target_heap,int*top){

    int change_key, priority;
    printf("\nenter the num to change priority:");
    scanf("%d",&change_key);
    int cur=IsExisted(target_heap,*top,change_key);
    printf("position of change key:%d\n",cur);
    if(cur==0)
    {
        printf("no such num existed\n");
        return;
    }
    printf("enter the priority to change to:");
    scanf("%d",&priority);
    element copy[MAX_SIZE];
    int arr[*top];
    int counter=*top;
    int cp_top=counter;
    for(int i=1;i<counter;i++){
        copy[i]=*(target_heap+i);
```

```
    }
    for(int i=1;i<counter;i++){
        arr[i]=RemoveMinHeap(&copy[0],&cp_top);
    }
      for(int i=1;i<counter;i++){
         printf("arr[i]=%d\n",arr[i]);
    }
    int prio_value=arr[priority];
    int prior_index;
    for(int i=1;i<counter;i++)
        if((target_heap+i)->key==prio_value){ prior_index=i;break;}

    int temp=(target_heap+cur)->key;
    (target_heap+cur)->key=(target_heap+prior_index)->key;
    (target_heap+prior_index)->key=temp;
};
```

說明：在決定欲更改優先權的值與優先權順位後，這個函式會先搜尋欲修改的數值的位置，若堆疊中不存在該值，則結束這個函式。當值存在，這個函式會計算欲修改數值於陣列中的位置，與該優先權順位原本的數值，以及其存在於陣列中的位置。 最後將兩者互換，達到修改優先順序的效果。(然而當執行了 `insert` 或 `remove` 後 順序可能會發生改變)