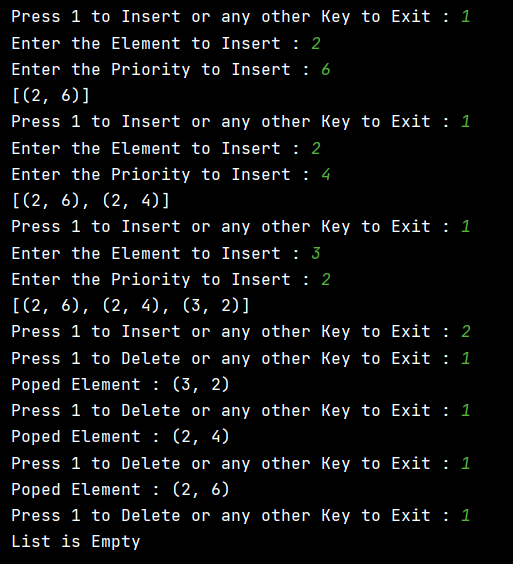
**S Abhishek AM.EN.U4CSE19147**

**Data Structures**

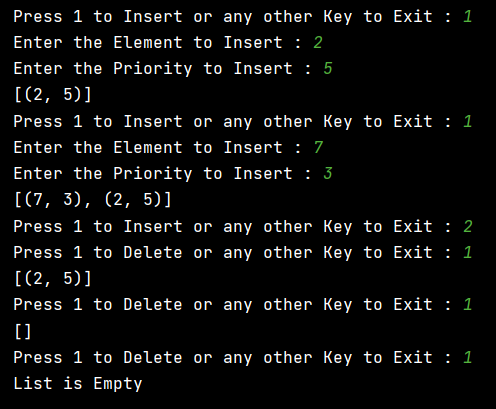
**1. Implement the priority queue using unsorted list.**

class list:  
 def \_\_init\_\_(self):  
 self.s = []  
 self.size = **0** def display(self):  
 if self.size == **0**:  
 print("Display : The List is Empty")  
 return  
 else:  
 print(self.s)  
  
 def enqueue(self):  
 while True:  
  
 x = int(input("Press 1 to Insert or any other Key to Exit : "))  
 if x == **1**:  
 a = int(input("Enter the Element to Insert : "))  
 b = int(input("Enter the Priority to Insert : "))  
 self.s.append((a**,** b))  
 self.size = self.size + **1** self.display()  
  
 else:  
 return  
  
 def dequeue(self):  
 while True:  
 x = int(input("Press 1 to Delete or any other Key to Exit : "))  
 if x == **1**:  
 if self.size == **0**:  
 print("List is Empty")  
 return  
 else:  
 self.s = sorted(self.s**,** key=lambda x: x[**1**])  
 temp = self.s.pop(**0**)  
 print("Poped Element : {}".format(temp))  
 self.size = self.size - **1** else:  
 return  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 l = list()  
 l.enqueue()  
 l.dequeue()



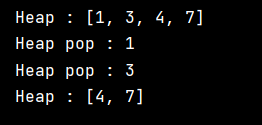
**2. Implement the priority queue using sorted list.**

class list:  
 def \_\_init\_\_(self):  
 self.s = []  
 self.size = **0** def display(self):  
 if self.size == **0**:  
 print("Display : The List is Empty")  
 return  
 else:  
 print(self.s)  
  
 def enqueue(self):  
 while True:  
  
 x = int(input("Press 1 to Insert or any other Key to Exit : "))  
 if x == **1**:  
 a = int(input("Enter the Element to Insert : "))  
 b = int(input("Enter the Priority to Insert : "))  
 self.s.append((a**,** b))  
 self.s = sorted(self.s**,** key=lambda x: x[**1**])  
 self.size = self.size + **1** self.display()  
  
 else:  
 return  
  
 def dequeue(self):  
 while True:  
 x = int(input("Press 1 to Delete or any other Key to Exit : "))  
 if x == **1**:  
 if self.size == **0**:  
 print("List is Empty")  
 return  
 else:  
 self.s.pop(**0**)  
 self.display()  
 self.size = self.size - **1** else:  
 return  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 l = list()  
 l.enqueue()  
 l.dequeue()



**3. Implement the priority queue using heap**

class Heap:  
 @classmethod  
 def push(cls**,** iterable**,** val):  
 iterable.append(val)  
 Heap.\_bubble\_up(iterable**, 0,** len(iterable) - **1**)  
  
 @classmethod  
 def pop(cls**,** iterable):  
 last\_item = iterable.pop()  
 res = iterable[**0**]  
 iterable[**0**] = last\_item  
 Heap.\_bubble\_to\_bottom(iterable)  
 return res  
  
 @classmethod  
 def \_bubble\_to\_bottom(cls**,** iterable**,** start=**0**):  
 end\_index = len(iterable)  
 current\_index = start  
 out\_of\_place\_item = iterable[current\_index]  
 child\_index = (**2** \* current\_index) + **1** while child\_index < end\_index:  
 right\_child\_index = (**2** \* current\_index) + **2** if right\_child\_index < end\_index and iterable[right\_child\_index] <= iterable[child\_index]:  
 child\_index = right\_child\_index  
  
 iterable[current\_index] = iterable[child\_index]  
 current\_index = child\_index  
 child\_index = (**2** \* current\_index) + **1** iterable[current\_index] = out\_of\_place\_item  
 Heap.\_bubble\_up(iterable**,** start**,** current\_index)  
  
 @classmethod  
 def heapify(cls**,** iterable):  
 for i in range((len(iterable) // **2**) - **1,** -**1,** -**1**):  
 Heap.\_bubble\_to\_bottom(iterable**,** i)  
  
 @classmethod  
 def \_bubble\_up(cls**,** iterable**,** stop**,** start):  
 current\_index = start  
 new\_item = iterable[current\_index]  
 while current\_index > stop:  
 parent\_index = (current\_index - **1**) >> **1** if new\_item < iterable[parent\_index]:  
 iterable[current\_index] = iterable[parent\_index]  
 current\_index = parent\_index  
 continue  
 break  
 iterable[current\_index] = new\_item  
  
a = []  
Heap.push(a**, 7**)  
Heap.push(a**, 4**)  
Heap.push(a**, 1**)  
Heap.push(a**, 3**)  
print("Heap : {}".format(a))  
print("Heap pop : {}".format(Heap.pop(a)))  
print("Heap pop : {}".format(Heap.pop(a)))  
print("Heap : {}".format(a))

****

**One Drive :** [**Click Me!!**](https://amritauniv-my.sharepoint.com/:f:/g/personal/sabhishek_am_students_amrita_edu/ElN2IRRv_MtGsZXhcmxt4XQBJfeeZVnEv5YF106zB7nVjQ?e=aya3ay)

**Thankyou**