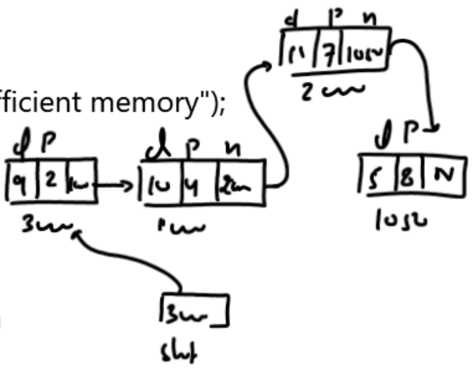
$$\begin{tikzcd}
& \underline{\text{Implement Priority Queue}} \arrow[dl] \arrow[dr] & \\
\text{Linked Rep} & & \text{Arry Rep}
\end{tikzcd}$$

```
struct PQueue
{
   int data;
   struct PQueue *next;
   int pr;
};
void insert(struct PQueue**,int,int);
void display(struct PQueue *);
int main()
{
   struct PQueue *start=NULL;
   insert(&start,10,4);
   insert(&start,5,8);
   insert(&start,11,7);
   ....
  display(start);
  reurn 0;
}
```

```
void insert(struct PQueue**ps,int dt,int pt)
{
   struct PQueue *p,*temp,*prev;
   p=(struct PQueue *)malloc(sizeof(struct PQueue));
   if(p==NULL)
   {
     printf("Insufficient memory");
     return;
   }
```



```
 p->data=dt;
 p->pr=pt;
 if(*ps==NULL)
  {
    *ps=p;
     p->next=NULL;
    return;
  }
  temp=*ps;
  while(temp!=NULL && temp->pr<pt)
  {
      prev=temp;
      temp=temp->next;
```

```
if(temp==NULL)
  {
    prev->next=p;
    p->next=NULL;
  }
else if(temp!=*ps)
{
   prev->next=p;
   p->next=temp;
}
else
{
   *ps=p;
   p->next=temp;
}
}
```

```
int del(struct PQueue**ps)
{
    struct PQueue *temp;
    if(*ps==NULL)
    {
       printf("Queue Underflow");
       return -1;
    }
 temp=*ps;
 int x=temp->pr;
 *ps=temp->next;
  free(temp);
  return x;
}
```

Array Impl of Priority Queue

(Assuming data is priority)

int arr[5];

values to be inserted : 4, 8, 2, 3 . . .

① insert(4)

0	1	2	3	4
4				

② insert(8)

0	1	2	3	4
4	8			

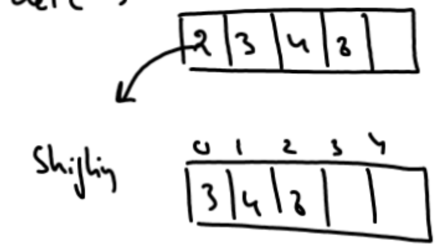
③ insert(2)

0	1	2	3	4
2	4	8		

④ insert(3)

0	1	2	3	4
2	3	4	8	

del( )



Implementation	insert	del	Access
① Linked List	O(n)	O(1)	O(1)
② Normal Array	O(n)	O(n) (shifting)	O(1)
③ Min heap	O(log n)	O(log n)	O(1)

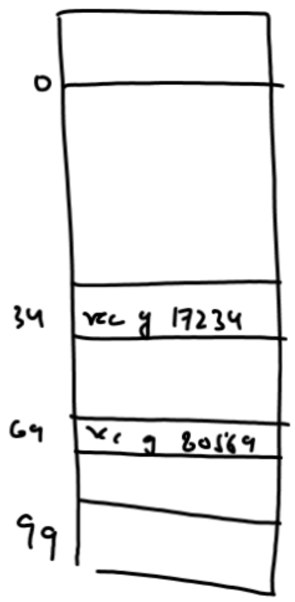
0 —— 99999

Hashing

```
struct Student
{   int roll;        90678
    char grade;
};  float per;
                     17234 % 100

struct Student s[100];   80569 % 100

                     10234 % 100
```



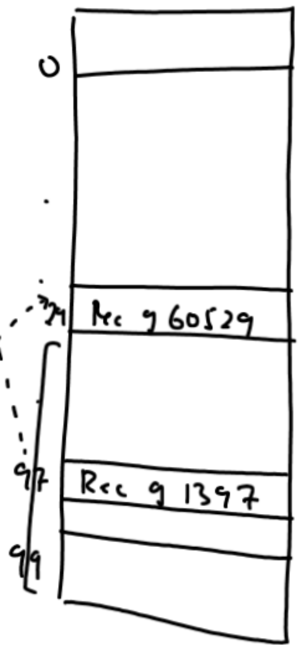
① Rec : 1397

Reduce : 1397 % 100 = 97

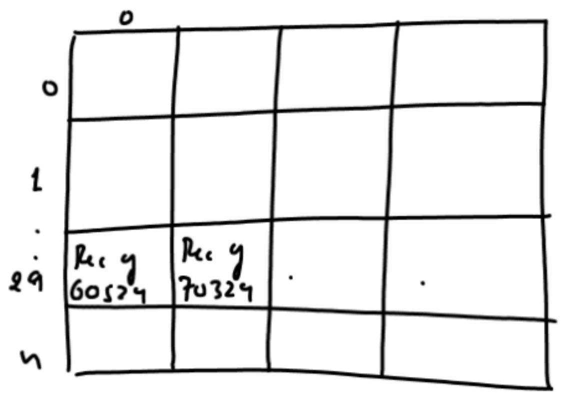
② Rec : 60529

Reduce : 60529 % 100 = 29

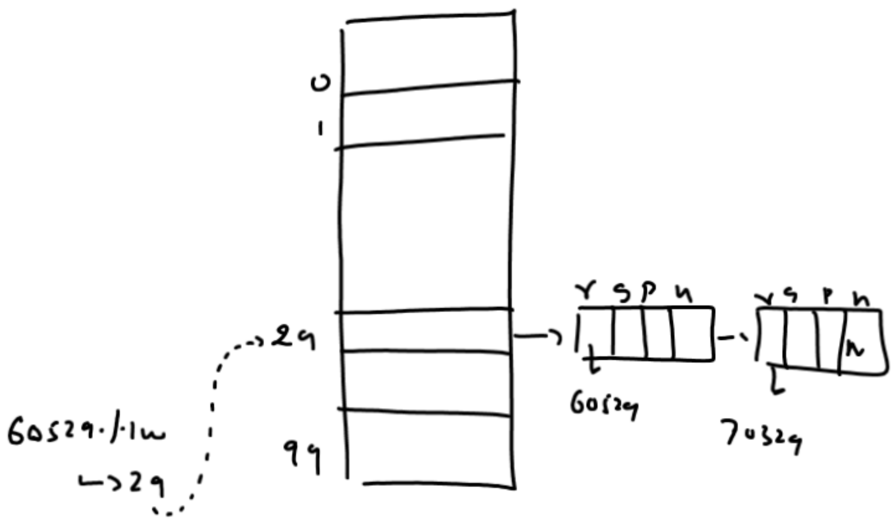
70329 % 100 → 29



```
int     linear_prb( int arr[ ], int index ,int size)
{
        while ( 1 )
        {
            index++;
            if (index == size)
                  index = 0;
            if ( arr[index] == -1)
                  retn index;
        }
}
```



struct Student * S[100];


0
1
29
99
60529 % 100
→ 29
r s p n
60529
r s p n
n
70329