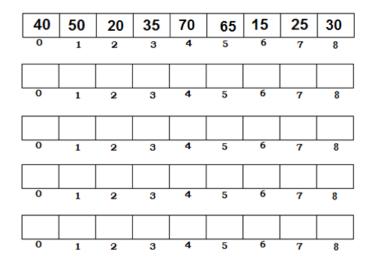
UniAcademia Engenharia de Software 2ª Chamada da 2ª Prova de Estrutura de Dados

Nome :	Nota :
Professor : Luiz Thadeu Grizendi	Data: 28/06/2022

1ª Questão - valor 3 pontos

Considere o método de ordenação QuickSort e o conjunto de chaves 40, 50, 20, 35, 70, 65, 15, 25 e 30. O algoritmo abaixo ilustra o procedimento partição. A finalidade deste procedimento é colocar o *pivot* no seu devido lugar.

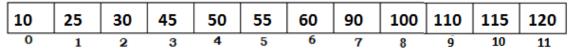
Mostrar passo a passo o algoritmo sendo executado e anotar os valores de **i** e **j** obtidos. Considere os parâmetros de entrada poMin = 0 e posMax = 8. A definição de estrutura de dados é a mesma utilizada na sala de aula.



UniAcademia Engenharia de Software 2ª Chamada da 2ª Prova de Estrutura de Dados

2ª Questão - valor 2 pontos

Considere o vetor abaixo ordenado pela chave. Quais são os endereços visitados (o que será impresso), considerando o as seguintes chaves 50 e 130, para os seguintes métodos:



a) Pesquisa sequencial (varredura ou scan)

```
# iterative scan search
def scan(alist, key):
    for i in range(0,len(alist)):
        print(i)
        if key == alist[i]:
            return i
        if alist[i]>key:
            return NOT_FOUND
return NOT_FOUND
```

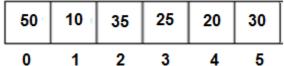
b) pesquisa binária

```
# iterative binary search
def binarySearch(alist, key):
  low = 0
  high = len(alist)-1
  mid=(low+high) //2
  while(low<=high):
    print(mid)
    if key == alist[mid]:
       return mid
    if key < alist[mid]:
       high=mid-1
    else:
       low=mid+1
    mid=(low+high) //2
  return NOT FOUND</pre>
```

UniAcademia Engenharia de Software 2ª Chamada da 2ª Prova de Estrutura de Dados

3ª Questão - valor 3 pontos

Considere o vetor a seguir. Informar qual é a sequência que será impressa após a execução dos métodos a seguir, supondo o vetor dado como entrada.



```
a)
   def bubbleSort(alist):
     N = len(alist)
      for passnum in range(1,N):
          trocou=False
          print('A')
          for i in range(0,N-passnum):
              if alist[i]>alist[i+1]:
                  sort.swap(alist,i,i+1)
                  print('B')
                  trocou=True
          if(not trocou):break
      return alist
b)
   def selectSort(alist):
     N = len(alist)
     for i in range (0, N-1):
        print('A')
        positionOfMin=i
        for j in range(i+1,N):
            if alist[j]<alist[positionOfMin]:</pre>
                positionOfMin = j
                print('B')
        sort.swap(alist,i,positionOfMin)
     return alist
c)
   def insertSort(alist):
     N = len(alist)
     for i in range(1,N):
       aux=alist[i]
       j=i-1
       print('A')
       # Achar a posição do menor elemento
       while( j >= 0 and alist[j] >= aux):
           alist[j+1]=alist[j]
           print('B')
           j=j−1
       alist[j+1]=aux
     return alist
```

UniAcademia Engenharia de Software

2ª Chamada da 2ª Prova de Estrutura de Dados

4ª Questão - valor 2 pontos

Considere o vetor abaixo ordenado, contendo dois vetores lógicos ordenados pela chave. O primeiro vetor, começa da posição 0, e termina na posição 3, o segundo vetor, começa na posição 4 e termina na posição 12. Executar o procedimento Merge, para gerar o arquivo final, anotando resultado que será impresso, sendo o valor da variável mid=4.

```
50 57
          35
             55
                   21
                       30 | 45
                                       80
                                           81
                                                88
     20
                                            10
                                                11
def merge(alist, mid):
      lefthalf = alist[:mid]
      righthalf = alist[mid:]
      i=0
      j=0
      k=0
      while i < len(lefthalf) and j < len(righthalf):</pre>
           if lefthalf[i] <= righthalf[j]:</pre>
               alist[k]=lefthalf[i]
               i=i+1
               print('A')
           else:
               alist[k]=righthalf[j]
               j=j+1
               print('B')
           k=k+1
      while i < len(lefthalf):</pre>
           alist[k]=lefthalf[i]
           i=i+1
           k=k+1
           print('C')
      while j < len(righthalf):</pre>
           alist[k]=righthalf[j]
           j=j+1
           k=k+1
           print('D')
      return(alist)
```

BOA PROVA!!!