Data Structures
ASSIGNMENT

D. ASHOK
142
CSE-C

1) Explain the phased procedure of Partition Exchange sort for the given list of elements and implement it. 51, 95, 66, 72, 42, 38, 39, 41, 15.

* Partition Exchange sort is also known as Quicksort Algorithm. The Basic Idea behind Quicksort is to partition the array into two subarrays, such that elements in subarray are less than equal to pivot element, elements in other sub array greater than pivot. This process is recursively applied to the subarrays until the entire array is sorted.

Program :-

```
#include<stdio.h>
int main()
{
  int i, size=9;
  int a[size]= {51, 95, 66, 72, 42, 38, 39, 41, 15};
  int n= sizeof(a)/sizeof(a[0]);
  quicksort(a[], 0, n-1);
  printf("Elements before sorting: \n");
    for(i=0; i<n; i++)
    {
      printf("%d\t", a[i]);
    }
  for(i=0; i<n; i++)
  {
    printf("%d", a[i]);
  }
}
int
{
```

Sorted list (2, 4, 8, 9) is 4+8/2.

```
int i= l+1 ;
int j =h;
while(i<=j)
{ while(a[i]<=pivot && i<=h)
  {
    i++;
  }
  while(a[j]>pivot && j>l)
  {
    j--;
  }
if(i<j)
{ int temp=a[i];
  a[i]= a[j];
  a[j]= temp;
  i++;
}  j--;
else i++;
}
int temp= a[j];
  a[j]=a[l];
  a[l]=temp;
return j;
}
void quicksort (int a[], int l, int h)
{ if(l<h)
  int p= partition(a,l,h);
   quicksort(a, l,p-1);
   quicksort (a,p+1,h);
  }
}
```

Output:

Before Sorting:

51 95 66 72 42 38 39 41 15

After Sorting:

15 38 39 41 42 51 68 72 95

2) Can we implement Queue Data Structure using Stack? if, yes explain the Procedure and give an Algorithm?

Yes, we can implement a queue using two stacks. The basic idea is to use one stack for the enqueue operation. Pushing an element into the queue and the other stack for the dequeue operation Popping elements from the front of the Queue. The two stacks

Algorithm :-

1) Start
2) Select enqueue, dequeue, peek, display, exit
3) In enqueue, if top == max - 1, print overflow, else stack[top] = item.
4) In dequeue, if top == -1 / print underflow, else Transfer all the element in stack1 to stack 2.
5) Then pop the elements in stack 2.
6) In [illegible] top == -1, print underflow, else transfer all the Elements [illegible] stack 1 to stack 2 then print the top most [illegible] stack 2.
7) [illegible] print all the element from stack[0] to
8) [illegible]

Explain the ~~Push~~ Procedure for following

Program:

```
#include <stdio.h>
#include <stdlib.h>
#define Max 10
int stack1[Max];
int stack2[Max];
int top1 = -1;
int top2 = -1;
void push(int value)
{
  if(top1 == Max-1)
  {
    printf("Queue full\n");
    return;
  }
stack1[++top1] = value;
}
void push2(int value)
{
  if(top2 == Max-1)
  {
    printf("Queue Underflow");
    return;
  }
stack2[++top2] = value;
}
```

Underflow");

```
    return -1;
  }
  return stack2[top2--];
}
void enqueue(int value)
{
  push(value);
}
int dequeue()
{
  if(top2==-1)
  {
    while(top1!=-1)
    {
      push2(pop1());
    }
  }
  return pop2();
}
int main()
{
  enqueue(1);
  enqueue(2);
  enqueue(3);
  printf("Dequeued:%d\n", dequeue());
  printf("Dequeued:%d\n", dequeue());
  enqueue(4);
  printf("Dequeued:%d\n", dequeue());
  printf("Dequeued:%d\n", dequeue());
  return 0;
}
```

1) Write a function to return the median value in a sorted linked list. If the length i of the list is odd, then the median is the ceiling(i/2) member, for example, given list (1,2,2,5,7,9,11) as input. your function should return the value 5. If the length of the list is even, then the median is the mean of the i/2 and i/2)+ member. Thus, the median of the sorted list (2,4,8,9) is 4+8/2.

finally, define the median of an empty list to be 0.

A)
```
#include <stdio.h>
#include <stdlib.h>
Struct Node
{
    int data;
    Struct node * link;
};
float findmedian (Struct Node *start)
{ if (start == NULL)
        return 0;
int mid = count/2 + 1;
 p = start;
for(i=1; i<mid; i++)
    p = p -> link;
if(count%2 != 0)
    return p -> data;
else
 return (p -> data + p -> link -> data)/2.0;
}
int main()
{
 Struct node * insertatbegin (Struct node * start, int data)
  {
      Struct node *temp = (Struct node*) malloc (sizeof (Struct node));
        temp -> info = data;
        temp -> link = start;
         start = temp;
        return start;
  } }
```

2) following elements are inserted into an empty hash function $f(x) = x \% 17$ and quadratic probing.

20, 10, 5, 30, 40, 57, 35, 25, 18, 22, 21

i) Draw the hash table for each insertion.

ii) What is the load factor after last insertion

iii) What is the maximum number of buckets examined in an unsuccessfull search?

Given Data: 20, 10, 5, 30, 40, 57, 35, 25, 18, 22, 21

$f(x) = x \% 17$

$f(20) = 20 \% 17 = 3$

$f(10) = 10 \% 17 = 10$

$f(5) = 5 \% 17 = 5$

$f(30) = 30 \% 17 = 13$

$f(40) = 40 \% 17 = 6$

$f(57) = 57 \% 17 = 6$ x

$f(x) = (f(x) + i^2) \% 17$

$= (6+1) \% 17 = 7$

$f(35) = 35 \% 17 = 1$

$f(25) = 25 \% 17 = 8$

$f(18) = 18 \% 17 = 1$

$f'(18) = (1+1) \% 17 = 2$

$f(22) = 22 \% 17 = 5$

$f'(22) = (5+1) \% 17 = 6$

$f'(22) = (5+4) \% 17 = 9$

$f(21) = 21 \% 17 = 4$

0	-1
1	35
2	18
3	20
4	21
5	5
6	40
7	57
8	25
9	22
10	10
11	-1
12	-1
13	30
14	-1
15	-1
16	-1

11) load factor:

$$\lambda = \frac{n}{m} = \frac{\text{no. of elements inserted}}{\text{table size}} = \frac{11}{17} = 0.647$$