Solution and Evaluation Scheme : T1 Examination ODD SEM 2021

Q1. To model a reservation system as an Abstract Data Type (ADT) specify the domain and set of operations.
Solution:
Domain: Seats – marks 0.5/0
Operations – marks 0.5/0
1)Determine available seats
2) Reserve a seat
3)Cancel a reservation
4)Find a block of available seats

Q2. int main()
```
 {
   char const *S1=" This is interesting";
   char *S2;
   _____________________
   strcpy(S2,S1);
   cout<<"S2="<<S2;
 }
```

Write the missing line in the code?

Solution : marks 1/0
S2=new char[strlen(S1)];

Q3. Write a function void reverseArray(int *a, int size) to reverse the elements of the array without using any additional variables or space to store elements. (do not use STL)
i/p array Arr =[1 ,2 ,3 ,5 8, 10]
o/p array Arr=[10,8,5,3,2,1]

Solution: marks 1/0

```
void reverseArray(int *arr,int size)
{
for (int i=0;i<size/2;i++)
{
arr[i]=arr[i]+arr[size-i];
arr[size]=arr[i]-arr[size-i];
arr[i]=arr[i]-arr[size-i];
}
```

Q4.Ram implemented a queue using a linked list. He additionally desires to delete the last element of the queue in constant time. Which data structure would be his optimal choice and Why?

Solution : marks 1/0
He needs to use a doubly-linked list, so that he can dequeue nodes from either end of the list in O(1) time.

Q5. A programmer performs a series of random push and pop operations using integers 0 through 9 in order on to the stack. Then he continues the same exercise with a series

of enqueue and dequeue operations on a queue. Comment whether the following options for order are correct or not and why?
(a) Stack : 4 3 2 1 0 9 8 7 6 5 , Queue: 0 1 2 3 4 5 6 7 8 9
(b) Stack : 2 1 4 3 6 5 8 7 9 0, Queue: 4 3 2 1 0 5 6 7 8 9
(c) Stack : 0 4 6 5 3 8 1 7 2 9, Queue: 0 1 2 3 4 5 6 7 8 9
(d) Stack : 4 6 8 7 5 3 2 9 1 0, Queue: 4 3 2 1 0 5 6 7 8 9

Solution : marks 1/0
a) Correct
b), c), d) Incorrect

Q6. Fill the blank spaces (represented by dotted lines) in the following code so output of the code is 20 (*i.e.* value of t is 20)

```
#include <iostream>
using namespace std;
int t;
void abc(int x, int y, int z)
{
   if(x < 1 || y < 1 || z < 1)
      return;
   else
   {
      t++;
      abc(x-2 , ___ , ____ );
      abc( ____ , ____ , z-2);
   }
}
int main()
{
   t = 0;
   abc(10, 20, 30);
   cout << t << endl;
   return 0;
}
```

Solution:

abc(x-2, y-2, z-3); marks 0.5/0
abc(x-3, y-1, z-2); marks 0.5/0

Q7. You are given an array A of size n and a value x. Explain the functionality of function F1.

```
vector<int> v;
int F2(int cal, int x, int n)
{
   int a1 = 0;
   int a2 = n;
   int m;
   int sol = -1;

   while(a1<= a2) {
      m = a1 + (a2–a1) / 2;
      if(v[m] - cal<= x) {
         a1 = m + 1;
         sol = m;
```

```
        }
 return sol;
}


void F1(int A[], int n, int x)
{
   int s = 0;
   v.push_back(0);

      for(int j = 0; j<n; j++) {

       s += A[j];
       v.push_back(s); }
       int a1 = 0, sol = 0, a2;
      for(int j = 0; j<x; j++) {

       a2 = F2(v[j], x, n);
       sol = max(sol, a2 - j);  }
       cout<<sol;
}
```

Solution: marks 1/0

The function F1 is finding the length of the longest sub-array having sum equal to the given value x using Binary Search.

Q.8 Add the line code in the mentioned dotted lines such that the function reverses the linked list.

```
Node * reverseList (LinkedList * l)
{
Node * a=l->head;
Node *b= l->head->next;
while(-------------)
{Node * c=b->next;
b->next=a;
---------;
b=c;
c=c->next;
}
b->next=a;
l->head->next=NULL;
----------;
return l->head;
}
```

Solution : marks 1/0

```
Node * reverseList (LinkedList * l)
{
Node * a=l->head;
Node *b= l->head->next;
while(b->next! =NULL)
{Node * c=b->next;
b->next=a;
a=b;
```

```
b=c;
c=c->next;
}
b->next=a;
l->head->next=NULL;
l->head=b;
return l->head;
}
```

Q9. Which type of hash function would be best suited to resolve the collisions in open addressing if two keys are starting from same hash address? Give the appropriate reason.
Solution: marks 1/0

Double hashing is best suited when the keys have same starting address and follow the same path in case of linear and quadratic probing.

Reason: If two keys starts from the same hash address, they both will follow same path in linear manner to skip 'i' full slots which creates primary cluster.
Similarly, in case of quadratic probing, secondary cluster will be created.

Q10. When can double hashing behave like linear probing?
Solution : marks 1/0
Linear probing is equivalent to double hashing with a secondary hash function of h2(k) = 1.

PART B

Q11. A singly linked list is created by inserting 8 nodes (N1, N2, N3, N4, N5, N6, N7 and N8) having following elements: 1, 2, 3, 4, 5, 6, 7, and 8. starting address of the linked list is stored into start pointer. And structure of a node of a singly linked list is defined as follows:

```
struct node
{
        int info;
        struct node *next;
};
```

Following recursive function is used to traverse the linked list which is called in main function as FUN1(start)

```
void FUN1(struct node *temp)
{
  if(temp -> next != NULL)
    FUN1(temp -> next);
  cout<<temp->info<<" ";
}
```

Convert the recursive function FUN1 into a non-recursive function (i.e. we need to remove the recursion) using minimum no. of stacks. Considering that the stack is implemented using array.

```
#include <iostream>

using namespace std;

struct node
{
        int info;
```

```
        struct node *next;
};

struct node *arr[8];        // stack implemented using array



void fun1(struct node *temp)                // given recursive function
{
        if(temp -> next != NULL)
                fun1(temp -> next);
        cout<<temp -> info<<" ";
}



int push(struct node *temp, int top)        //push operation of stack
{
    arr[++top] = temp;
    return top;
}


int pop(int top)            //pop operation of stack
{
   cout<<arr[top--] -> info<<" ";
   return top;
}

void fun2(struct node *temp)                //removal of recursion using stack
{
   int top = -1;
   while(temp != NULL)
   {
      top = push(temp, top);
      temp = temp -> next;
   }
   while(top >= 0)
   {
      top = pop(top);
   }
}

int main()
{
        struct node *start = NULL, *temp, *last;
        int i = 1;
        while(i <= 8)
        {
                temp = new struct node;
                temp -> info = i;
                temp -> next = NULL;
                if(start == NULL)
          start = last = temp;
       else
```

```
        {
            last -> next = temp;
            last = temp;
        }
        i++;
            }

    fun1(start);
    cout<<"\n";
    fun2(start);
    return 0;
}
```

// in question, only recursion removal function was asked, so no main function is expected. It is there for our own reference
// Award 1 marks for removal of recursion function (in this code it is fun2)
// Award 0.5 marks each for push and pop operations

Q12. Amit organized a birthday party. The children played the "passing the parcel" game. The parcel is passed as fast as possible to the next child. It stops when the music ends after a fixed time interval and the child with the parcel has to perform an act given by the rest of the children. This repeats with the remaining children till all but one perform. The last one who doesn't perform is declared the winner. Suggest an efficient data structure to implement the real scenario. Write a pseudocode to implement the scenario.

Assume the fixed time slot when music stops allows m exchange of the parcel. The best data structure is circular linked list and problem is similar to Josephus problem where we assume-

```
void getJosephusPosition(int m, int n)
{
    // Create a circular linked list of
    // size N.
    Node *head = newNode(1);
    Node *prev = head;
    for (int i = 2; i <= n; i++)
    {
        prev->next = newNode(i);
        prev = prev->next;
    }
    prev->next = head; // Connect last
                       // node to first

    /* while only one node is left in the
    linked list*/
    Node *ptr1 = head, *ptr2 = head;
    while (ptr1->next != ptr1)
    {
        // Find m-th node
        int count = 1;
        while (count != m)
        {
            ptr2 = ptr1;
            ptr1 = ptr1->next;
```

```
            count++;
        }

        /* Remove the m-th node */
        ptr2->next = ptr1->next;
        ptr1 = ptr2->next;
    }

    printf ("Last person left standing "
            "(Josephus Position) is %d
",
            ptr1->data);
}
```

0.5 for specifying circular linked list
1.5 marks for pseudo code

Q13. Consider the 7 numbers - 2, 25, 35, 39, 40, 47, and 50. Show the steps to search key=34 using binary and interpolation search. Comment on no. of comparison required in searching the key in both cases.

Solution:
Search Key=34
Interpolation Search : 34
Here n = 7 Key = 34
low = 0
high = n – 1 = 6
mid = 0 + (6 – 0) × ((34 – 2)/(34 – 2))
= 6 × (32/48)
= 4
if(key < A[mid])
=> key < A[4]
=> 34 < 40
so reset high = mid–1
=>3
low = 0
high = 3
Since(low < high)
mid = 0+(3–0) × ((34–2)/(39–2))
= 3 × (32/37)
= 2.59 Here we consider only the integer part of the mid
i.e., mid = 2
if (key < A[mid])
=> key < A[2]
=>34 < 35
so reset high = mid–1
=> 1
low = 0
high = 1
Since (low < high)
mid = 0+(1–0) × ((34–2)/(25–2))
= 3 × (32/23)
= 1
here (key > A[mid])
=>key > A[1]

=> 34 > 25
so reset low = mid+1
=> 2
low = 2
high = 1
Since (low > high)
DISPLAY " The key is not in the array"
STOP

no. of Comparisons= 3
Binary search
Low =0 high=6 mid=(0+6)/2=3
A[mid]=39 >34
Drop the higher array
A={2,25,35}
Mid=1
A[mid]=25 <34 , lower array is dropped
A={35} ≠34, hence search key not found
# No. of Comparisons=3

Q14. You are given the following code to search an element in a given array. But the code has some logical problems.

a) In which cases the program will generate erroneous output?
b) Rectify the problem and change the lines in the given code to handle the above mentioned test case by you.

```
1.  Find(int m, int A[]) {
2.  int p, q, r;
3.  p= 0; q = 10;
4.  do {
5.  r = (p + q) /2;
6.  if(A[r] <m)
7.  p = r;
8.  else
9.  q = r;
10. } while(A[r]!=m&&p<q);
11. if(A[r] == m)
12. cout<<"m found ";
13. else
14. cout<<"m not found ";
15. }
```

Solution: The program fails when we search the last element of the array or element to be searched is greater than all the elements of the array. In those cases, while condition never becomes false, and program goes in an infinite loop. The problem can be resolved by replacing Line # 6-9 with the following code.

```
if(A[r] < m)
        p = r+1;
else
    q = r-1;
```

Q 15. Let us consider a hash table of size 11, having starting index zero, and a hash function h=(3x+ 5) mod7. Assuming the hash table is initially empty, Show the insertion steps and final hash table when the sequence 1, 5, 8, 13, 17 is inserted into the table using closed hashing?
Marks : completely correct : 2/error in computation : 1/0

Step 1: Initially, hash table is empty

values - - - - - - - - - - -

Slot 0 1 2 3 4 5 6 7 8 9 10

The value of function (3x + 5)mod 7 for 1 is 1, so let us put the value at 1

- 1 - - - - - - - - -

0 1 2 3 4 5 6 7 8 9 10

The value of function (3x + 5)mod 7 for 5 is 6,

- 1 - - - - 5 - - - -

0 1 2 3 4 5 6 7 8 9 10

The value of function (3x + 5)mod 7 for 8 is 1, but 1 is already occupied, let us put the value(8) at next available space(2)

- 1 8 - - - 5 - - - -

0 1 2 3 4 5 6 7 8 9 10

The value of function (3x + 5)mod 7 for 13 is 2, but 2 is already occupied, let us put the value(13) at next available space(3).

- 1 8 13 - - 5 - - - -

0 1 2 3 4 5 6 7 8 9 10

The value of function (3x + 5)mod 7 for 17 is 0,

17 1 8 13 - - 5 - - - -

0 1 2 3 4 5 6 7 8 9 10
