Answer to the question no 01

1.(a) $O(\sqrt{n})$

1.(b) $log_2(log_2n)$

3.(c)(log_2n)

Answer to the question no 02

STEP :

Before sort : 3 3 1 7 7 4 4 5

Maximum : 7

Frequency : 0 1 0 2 2 1 0 2

Cumulative Sum : 0 1 1 3 5 6 6 8 (i=i+1)

After sort : 1 3 3 4 4 5 7 7

The reason of backward traversing :

After Traversing the Original array , we prefer from last Since we want to add Elements in their proper position so when we subtract the index , the Element will be added to lateral position.
But if we start traversing from the beginning , then there will be no meaning for taking the cumulative sum since we are not adding according to the Elements placed. We are adding hap -hazardly which can be done even if we do not take their cumulative sum.

```
#include<iostream>
#define MAX 255
using namespace std;

void countSort(int array[], int size) {
   int output[MAX];
   int count[MAX];
   int max = array[0];
```

```
    // Step-01 Finding Max
    for (int i = 1; i < size; i++) { if (array[i] > max)
            max = array[i];
    }

    // step-02 Initialize of array to 0
    for (int i = 0; i <= max; ++i) {
        count[i] = 0;
    }

    // step-03 Frequency Calculation
    for (int i = 0; i < size; i++) {
        count[array[i]]++;
    }

    // step -04 Store the cummulative sum
    for (int i = 1; i <= max; i++)
           {
                      count[i] += count[i - 1];
           }

           // step-05 Final array --> Backword Travarsal of Basic array
           for (int i = size - 1; i >= 0; i--)
    {
        output[count[array[i]] - 1] = array[i];
        count[array[i]]--;
    }
    for (int i = 0; i < size; i++) {
        array[i] = output[i];
    }
}

// print an array
void display(int array[], int size) {
    for (int i = 0; i < size; i++)
        cout << array[i] << " ";
    cout << endl;
}
```

```
int main() {
    int array[] = {3,3,1,7,7,4,4,5};
    int n = sizeof(array) / sizeof(array[0]);

    countSort(array, n);

    display(array, n);

    return 0;
}
```

Answer to the question no 03

STEP 1:

Here upper bound 9 and lower bound 0 , so the mid is (0+9)/2=4.5

Mid : 4

12 is greater than 4 , 4 is on the left side. Because the array is sorted in ascending order.

STEP 2:

Here upper bound 4 and lower bound 0 , so the mid is (0+3)/2=1.5

Mid : 1

2 is less than 4 , 4 is on the right side. Because the array is sorted in ascending order.

Step 3:

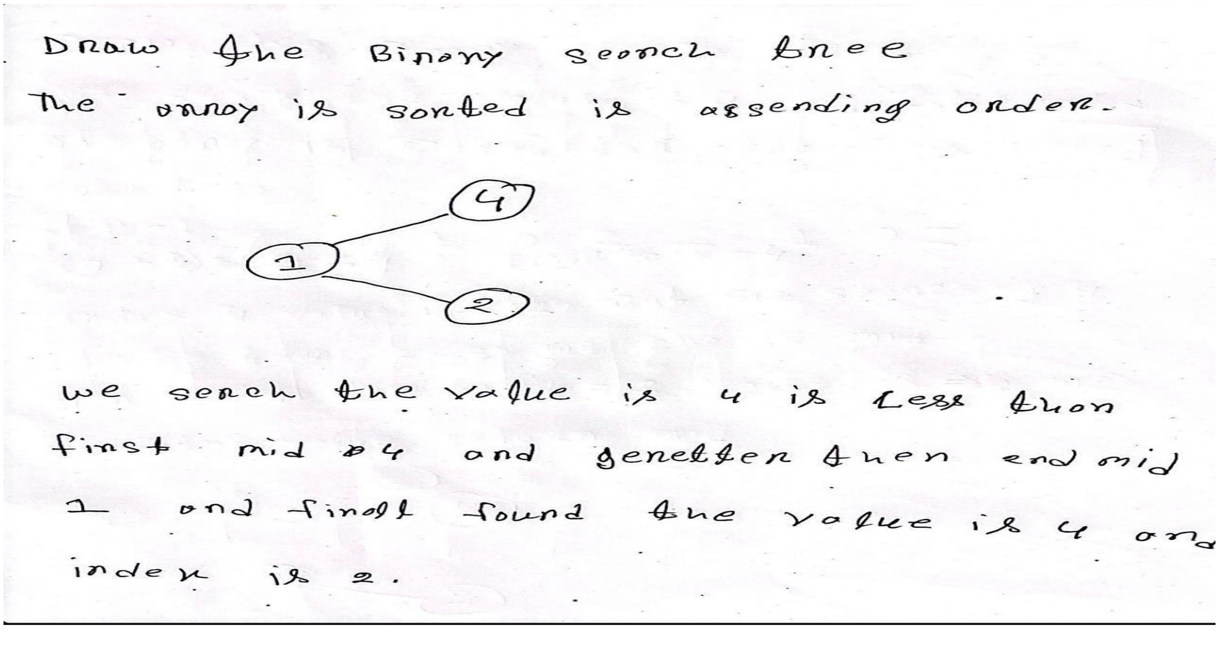
Here upper bound 3 and lower bound 2 , so the mid is (3+2)/2=2.5

So,mid is = 2,

We found our desired searching value is 4 in index number 2 so, return the mid .

```
#include <bits/stdc++.h>
using namespace std;
int binarySearch(int arr[], int p, int r, int num) {
   if (p <= r) {
      int mid = (p + r)/2;
```

```
        if (arr[mid] == num)
            return mid ;
        if (arr[mid] > num)
            return binarySearch(arr, p, mid-1, num);
        if (arr[mid] < num)
            return binarySearch(arr, mid+1, r, num);
    }
    return -1;
}
int main(void) {
    int arr[] = {1, 2,4,9,12,14,16,21,32,35};
    int n = sizeof(arr)/ sizeof(arr[0]);
    int num;
    cout << "Enter the number to search: \n";
    cin >> num;
    int index = binarySearch (arr, 0, n-1, num);
    if(index == -1){
        cout<< num <<" is not present in the array";
    }else{
        cout<< num <<" is present at index "<< index <<" in the array";
    }
    return 0;
}
```



Draw the Binary search tree

The array is sorted is assending order.

we serch the value is 4 is Less than first mid ~~8~~ 4 and greatter then end mid 1 and finall found the value is 4 and index is 2.

Answer to the question no 04

Arr[70][65]
Given that arr[0][0] = 1230.

Now we Find out the location of arr[3][18]

= 1230+4[65(3-0)+(18-0)
= 1230+4[195+18]
= 1230+4(213)
= 1230+852
= 2082

Answer to the question no 05

(a)28

(b)53

(c)Null

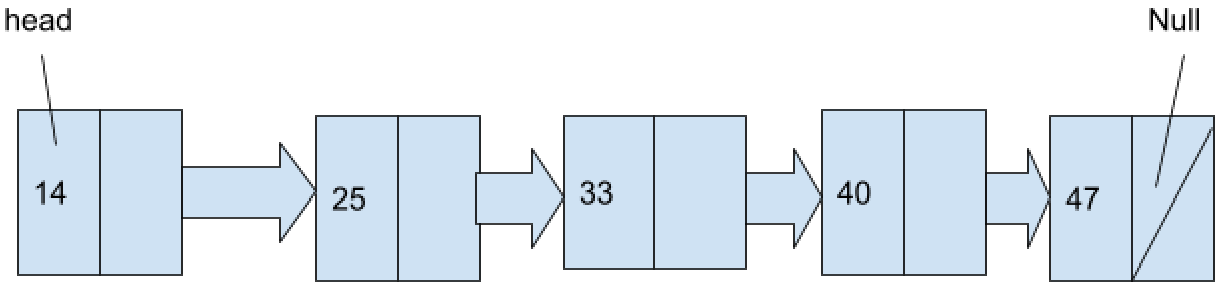
(d)16

(e)25

Answer to the question no 06

(a) We insert the value 40 between 33 and 47 we pass the position and we pass the value .

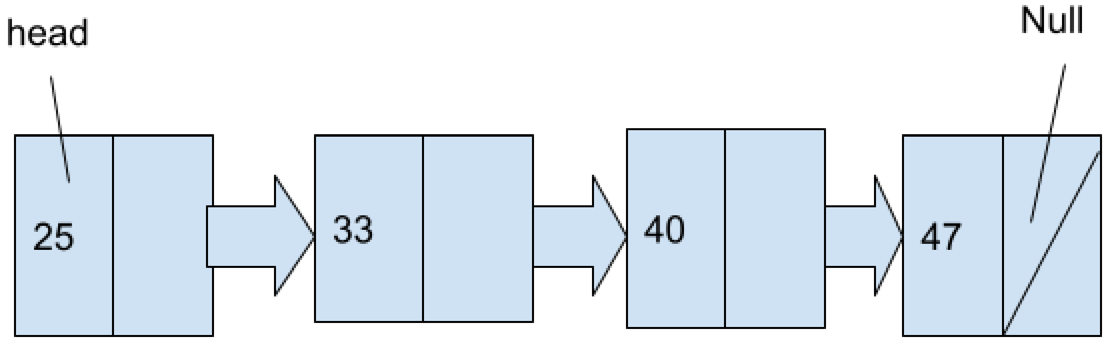
```
void insertSpecificPosition (Node * &head , int pos, int val){
  int i =0;
  Node * temp =head;

  while (i<pos-2){
     temp=temp->Next;
     i++
  }
  Node * newNode = new Node(val);
  newNode->Next =temp-> next;
  temp->Next = newnode;
}
```



(b)

14 is head of the list if we delete the head the we write this code

```
void deletion_at_head(Node* &head)
{
   Node* temp = head;
   if(temp!=NULL)
   {
      head = temp->next;
      delete temp;
   }

   else
   {
      cout<<"There is no value in the linked list"<<endl;
   }
}
```



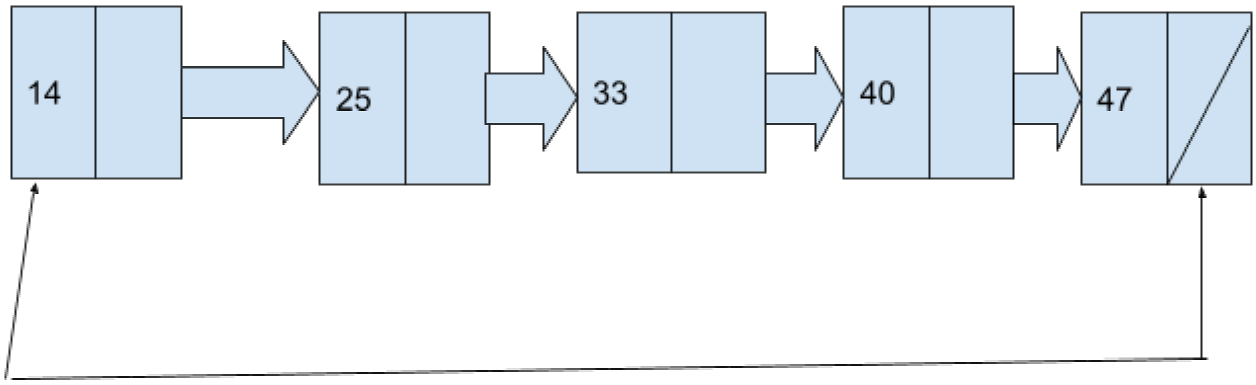
(c)

Make cycle

```
void make_cycle(Node* &head, int pos)
{
   Node* temp = head;
   Node* startNode = head;
   int count = 1;

   while (temp->next != NULL)
   {
      if(count == pos ) startNode = temp;
      temp = temp->next;
      count++;
   }

   temp->next = startNode;
}

bool detect_cycle(Node* &head)
{
   Node* fast = head;
   Node* slow = head;
   while(fast != NULL && fast->next != NULL)
   {
      fast = fast->next->next;
      slow = slow->next;
      if(slow->next == fast->next)
      {
         return true;
```

```
        }
    }
    return false;

}
```

We make a linear circular linked list from the current list



Answer to the question no 07

Write an algorithm to display the data stored in a doubly linked list in reverse order.

```
void reverse(Node** head_ref)
{
    Node* temp = NULL;
    Node* current = *head_ref;


    while (current != NULL) {
        temp = current->prev;
        current->prev = current->next;
        current->next = temp;
        current = current->prev;
    }
    if (temp != NULL)
        *head_ref = temp->prev;
}
```

Full code

```
#include <bits/stdc++.h>
using namespace std;


class Node {
public:
    int data;
    Node* next;
    Node* prev;
};


void reverse(Node** head_ref)
{
    Node* temp = NULL;
    Node* current = *head_ref;


    while (current != NULL) {
        temp = current->prev;
        current->prev = current->next;
        current->next = temp;
        current = current->prev;
    }


    if (temp != NULL)
        *head_ref = temp->prev;
}
```

```
void push(Node** head_ref, int new_data)
{

    Node* new_node = new Node();


    new_node->data = new_data;


    new_node->prev = NULL;


    new_node->next = (*head_ref);


    if ((*head_ref) != NULL)
        (*head_ref)->prev = new_node;

    /* move the head to point to the new node */
    (*head_ref) = new_node;
}


void printList(Node* node)
{
    while (node != NULL) {
        cout << node->data << " ";
        node = node->next;
    }
}


int main()
{

    Node* head = NULL;


    push(&head, 2);
    push(&head, 4);
    push(&head, 8);
    push(&head, 10);

    cout << "Original Linked list" << endl;
    printList(head);


    reverse(&head);

    cout << "\nReversed Linked list" << endl;
    printList(head);

    return 0;
}
```

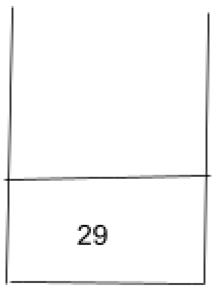
Answer to the question no 08

My Birthday is 28/03/1998 So, last digit of my birthday is 8

X = 8+5 = 13 , Y = 13+3 = 16, Z = 16+13 = 29

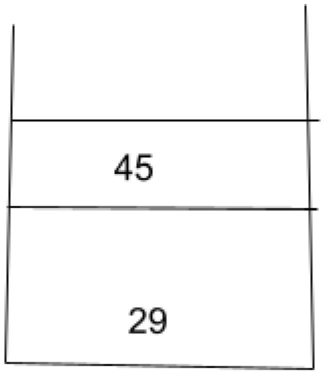
So, I Show the status of a STACK

Step -01 Push(x+y) = 13+16 =29



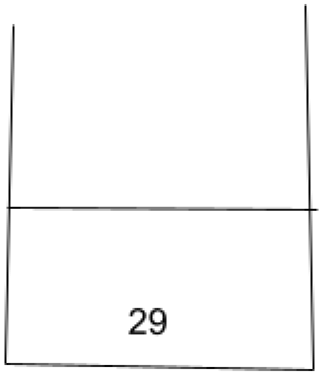
After push Stack

step -02 Push(y+z) = 16+29 = 45



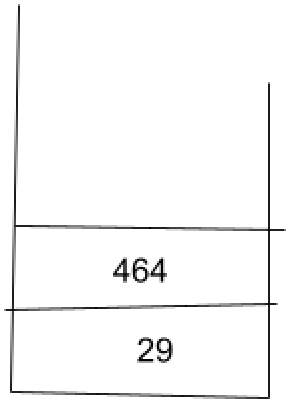
After push Stack

Step-03 Pop()



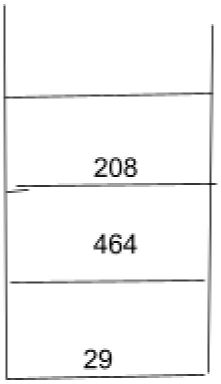
After pop Stack

Step -04 Push(y*z) Y*Z= 16*29 = 464



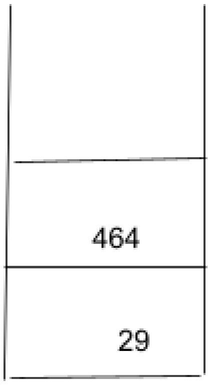
After push Stack

Step -05 Push(x*y) X*Y = 208



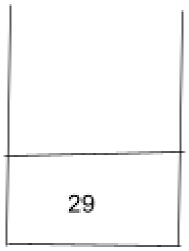
After push Stack

Step-06 Pop()



After pop Stack

Step -07 pop ( )



Final stack

Answer to the question no 09

1.

```
#include<stdio.h>
#include<string.h>
int top=-1;
char Stack[4]={‘\0’};
int main()
{
    char Str1[4]={‘\0’};
    char Str2[4]={‘\0’};
```

There have a 2 character array this name is str1 str2 both of character array space have 3

```
    int i;
    strcpy(Str1, “CSE”);
```

Cse string copy are str1

```
    for(i=0; i<3; ++i){
    Push(Str1[i]);
    }
```

This for loop use basically CSE string push in str1 stack sequentially like that

C
S
E

But str1 have no space and str1 have already CSE by calling function strcpy(Str1, “CSE”);

```
    for(i=0; i<3; ++i){
    Str2[i]=Pop();
    }
```

This for loop use basically CSE string pop in str2 stack sequentially but str2 have no character so pop function doesn't work str2 have no element

```
    printf(“%s”, Str2);
    return 0;
}
```

Str2 have no element so nothing print

```
void Push(char x){
      Stack[++top]=x;
      return;
}
```

This function basically changes the top character in the stack. Suppose stack top has B but we use this function and set the top element D.

```
char Pop(void){
   return Stack[top--];}
```

This function basically deletes the top character in the stack. Suppose stack top has B but we use this function

```
and set the top element previous. And sequentially delete
the top element.
```

Answer to the question no 10

If we work in array we face under overflow and overflow problem its basically depend array size if array size is 10 but we push 11 value we can't push 11 no value because it's over flow on the other hand array size is 0 we can’t push any value in thi -1 position we face the problem it’s call under overflow

Overflow code

```
#include <bits/stdc++.h>
using namespace std;

int addOvf(int* result, int a, int b)
{
    if (a >= 0 && b >= 0 && (a > INT_MAX - b)) {
        return -1;
    }

    else if (a < 0 && b < 0 && (a < INT_MIN - b)) {
        return -1;
    }
    else {
        *result = a + b;
        return 0;
    }
}

int main()
{
    int a, b;
    int* res;
    a = INT_MAX;
    b = 8192;
    cout << addOvf(res, a, b);

    return 0;
}
```

Under over flow code

```
v oid pop (){

        Q chk;
        if(rear == NULL){
            cout<<"Que is underflow && Que have no element "<<endl;
            return chk;
        }

        Node <Q>* dellNode;

        dellNode = front;
        front = front->next;
        if(front == NULL){
            rear ==NULL;
        }
        chk = dellNode->value;
        delete dellNode;
        return chk;
```

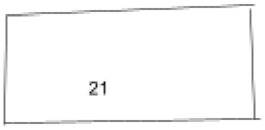
Answer to the question no 11

My Birthday is 28/03/1998 So, last digit of my birthday is 8

X = 9   , Y = X+3 = 12, Z = X+Y = 21, P = Y+Z = 33

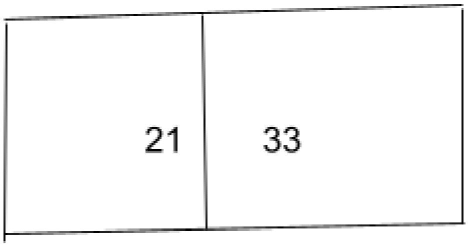
So, I Show the status of a Queue

Step -01   Enqueue(z)= 21



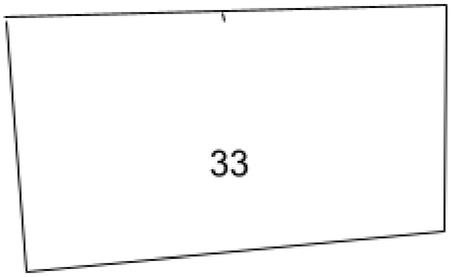
After Enqueue(z)

step -02 Enqueue(p) = 33



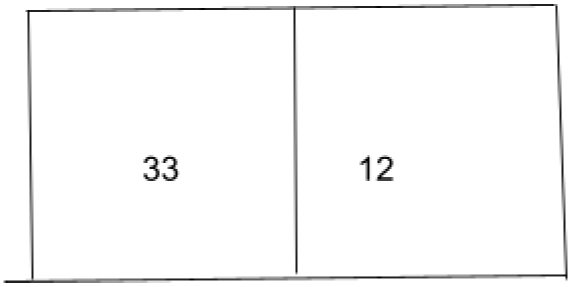
After Enqueue(p)

Step-03 Dequeue()



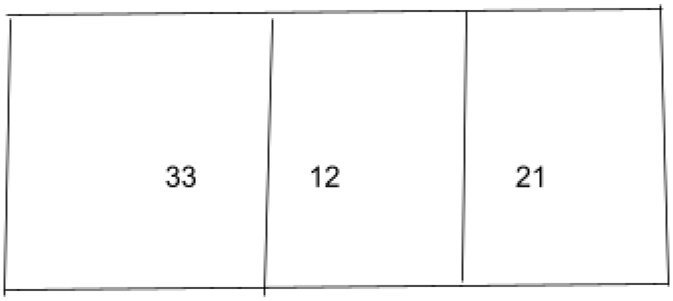
After Dequeue()

Step -04   Enqueue(y) = 12



After Enqueue(y) = 12

Step -05 , Enqueue(z) = 21



After Enqueue(z)

Answer to the question no 12

```
#include <bits/stdc++.h>
using namespace std;

int sum(int arr[], int n)
{
   int sum = 0;


   for (int i = 0; i < n; i++)
   sum += arr[i];

   return sum;
}


int main()
{
   int n;
   cin>>n;
   int arr[n];

   for ( int i=0; i<n; i++){
      cin>>arr[i];
```

```
    }
    cout << "Sum of given array is " << sum(arr, n);
    return 0;
}
```