Exp. 2.

Q1)

Ans.

```
#include<stdio.h>
int main() {
int a[] = {56, 36, 89, 57, 01, 00, 67, 59}
int k, j. found = 0;
printf("Enter key to be search);
scanf("%d", &k);
for (i=0; i<8; i++)
{
   if (a[i] == k)
{
 printf("\n key found At index: %d; i);
 found = 1;
  break;
 }
}
 if (found == 0)
 printf("\n key not found in Array");
 return 0;
}
```

Output :- ~~Enter~~ number of search : 01
Element found in the array

Ans

```
#include <stdio.h>
int main()
{
  int A[100], n, m, l, h, i
  printf("Enter number of Array elements: ");
  scanf("%d", n);
  printf("Enter key to be seached: ");
  scanf("%d", k);
  printf("\n Enter Sorted Array elements: \n");
  for (i=0; i<n; i++)
  {
    scanf("%d", a[i]);
  }
  l = 0; h = (n-1);
  while(l<=h)
  {
    m = (int)(l+h/2);
    if(n = a[m])
  {
    Printf("\n element found at index: %d",m
    break;
  }
  else
    if(n > a[m])
  {
    l = m+1;
  }
  if(n < a[m];
  {
    h = m-1;
  }
  }
```

```
}
  if (n<1)
  printf("\n Element not found");
  return 0;
}
```

Output

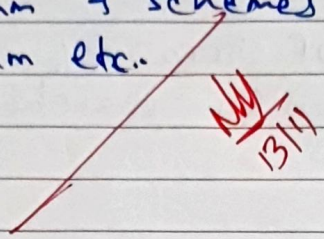
Enter Number of Array Element = 6
Enter key to be searched : 23
23
48
67
56
78
98

Q3)

Ans. linear search	Binary search
i) Time complexity is low.	Time complexity is O.
ii) Works on both sorted and Converted.	Only works on sorted data.
iii) It is less efficient especially for large data sets.	It is more efficient especially for large data sets.
iv) Code is simple.	Code is complex
v) It use sequential searching Approach.	It uses divide & conquer approach.

Q4

Ans A linear search runs in an worts linear time & makes at most comparision where n is the long the of the list. Linear Search is rarely practical ~~became~~ because other search Algorithm & schemes such as binary Search Algorithm etc..



Q) Find the following patterns.

i)
```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

→
```
#include <stdio.h>
int main()
{
  int i, j;
for (i=0; i<=5; j++)
{
  for (j=1; j<=i; j++){
  printf("%d", i);
}
return 0;
}
```

(i)

```
1
1 2
3 3 3
1 2 3 4
5 5 5 5 5
```

->

```
#include<stdio.h>
int main()
{
int i,j;
for(i=1;i<=5;i++)
{
if(i%2==0)
{
for(j=1;j<=i;j++)
{
printf("%d",j);
}
}
else{
for(j=1;j<=i;j++)
{
printf("%d",i);
}
printf("\n");
}
return 0;
}
```

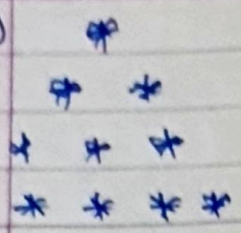
(iii)

```
*
# #
$ $ $
? ? ? ?
```

⇒

```
#include<stdio.h>
int main()
{
int i,j;
for(i=0;j<=4;i++)
{
switch(i)
{
case 1:
printf("*\n");
break;
case 2:
printf("# #\n");
break;
case 3:
printf("$$$\n");
break;
default:
break;
}
}
return 0;
}
```

iv)



→

```
#include <stdio.h>
  int main()
{
   int i,j, space;
   int rows = 4;
  for (i=1; i<=rows; i++)
  {
   for (space = 1; space <= rows - i; space++)
   {
     printf(" ");
   }
    for (j = 1; j<=i; j++){
     printf("* ");
   }
   return 0;
  }
```

VJ

```
* * * *
 * * *
  * *
   *
```

→

```
#include<stdio.h>
int main(){
 int i, j, space;
 int rows = 4;
 for (i=0; i<rows; i++){
 for (space = 0; space<i ; space++){
  printf(" ");
 }
  printf(" ");
{
 for (j = 0; j<rows-i ; j++)
{
  printf("* ");
{
 printf("\n");
  }
 return 0;
}
```