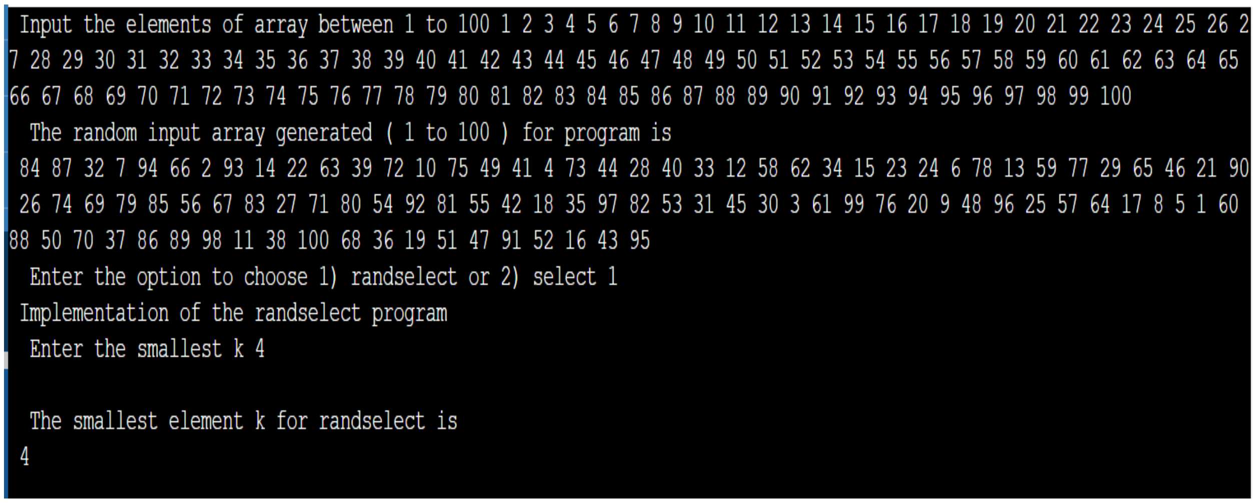
**** Rand-Select (with linear expected running time) and Select (with linear worst-case running time)**

```
#include <iostream>
#include <cstdlib>
#include<ctime>
#include<algorithm>
#include<climits>
using namespace std;

int partition(int arr[], int first, int last,int pivot)
{
    if(pivot==0)
     pivot = arr[last]; //taking the pivot

    int a = (first - 1);  // taking the lower element

    for (int b = first; b <= last - 1; b++)
    {
        if (arr[b] <= pivot)
        {
            a++;
            swap(arr[a], arr[b]);
        }
    }
    swap(arr[a + 1], arr[last]);
    return (a + 1);
}


int partition_r(int arr[], int first, int last)
{
    // Taking the random index using srand such that it sets the starting point of producing the series of
random numbers generated.
    srand(time(NULL));
    int random = first + rand() % (last - first); //using rand function to generate random numbers

    // Swaping the index with last element as per this algorithm
    swap(arr[random], arr[last]);

    return partition(arr, first, last,0);
}

// for randselect i is the smallest element selected
int randselect(int arr[], int p, int q,int i )
```

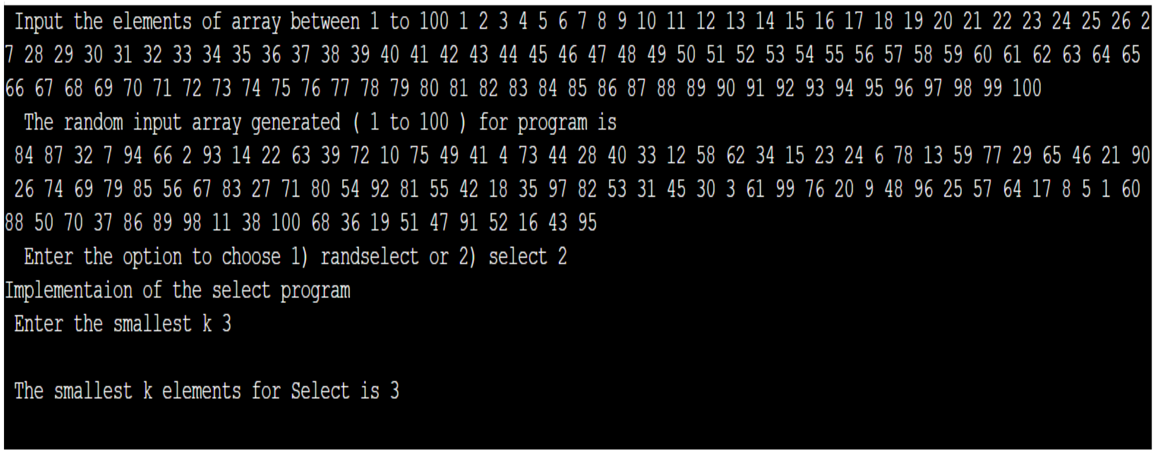
```
{
 int r,k;
 if(p==q)
 return arr[p];
 else

      r=partition_r(arr,p,q);
       k=r-p+1;
       if (i==k)
       return arr[r];
       else
            if(i<k)
            return randselect(arr,p,r-1,i);
            else
            return randselect(arr,r+1,q,i-k);
}

// Function to return the middle element of the array
int findM(int arr[], int size)
{
   sort(arr, arr+size);
   return arr[size/2];
}
// generating in the worst case linear time senario
int Select(int arr[], int l, int r, int k)
{

   if (k > 0 && k <= r - l + 1)
   {
      int n = r-l+1; //no of elements

      // Divide n elements into group of 5 and finding medians
      int i, median[(n+4)/5];
      for (i=0; i<n/5; i++)
         median[i] = findM(arr+l+i*5, 5);
      if (i*5 < n)
      {
         median[i] = findM(arr+l+i*5, n%5);
         i++;
      }

      // Find median of all medians using recursive call.
      int MOfM = (i == 1)? median[i-1]:
                         Select(median, 0, i-1, i/2);
```

```
        // Partition the array with x as a pivot
        int pos = partition(arr, l, r, MOfM);

        // If position is same as k
        if (pos-l == k-1)
            return arr[pos];
        if (pos-l > k-1)  //randomly select the i smallest elemet in the part of array
            return Select(arr, l, pos-1, k);

        // Else recursively select the i-k smallest in the upper part of the array .
        return Select(arr, pos+1, r, k-pos+l-1);
     }

   return INT_MAX;  // if k ecxeeds the limits
   //return k;
}
void generaterandomarray(int A[])
{
    int d,l,h,e;
    int m;
   for(int i=0;i<=99;i++) // function to generate random numbers
   {
       int random = (rand() % 100) ;
       swap(A[i],A[random]);
   }
      cout<<" The random input array generated ( 1 to 100 ) for program is ";
      cout<<" \n ";
      for(int s=0;s<=99;s++)
      cout<<A[s]<<" ";
      cout<<" \n ";
      cout<<" Enter the option to choose 1) randselect or 2) select ";
      cin>>m;
      switch(m)
       {
      case 1:
              cout<<" Implementation of the randselect program ";
              cout<<" \n ";
              cout<<" Enter the smallest k ";
              cin>>e;
              cout<<" \n ";
              h=randselect(A, 0,99,e);
              cout<<" The smallest element k for randselect is ";
              cout<<" \n ";
              cout<<h; // since the index started with zero
              cout<<"\n";
              break;
      case 2:
              cout<<"Implementaion of the select program ";
              cout<<"\n";
              cout<<" Enter the smallest k ";
              cin>>d;
              cout<<"\n";
              Select(A,0,99,d);
              l= Select(A,0,99,d);
              cout<<" The smallest k elements for Select is "<<l; //since the index started with zero
              break;
      default: cout<<" Choose correct option ";
             break;
       }
}
int main() {
        int c[500];
        cout<<" Input the elements of array between 1 to 100 ";
        for(int i=0;i<=99;i++)
        {
          c[i]=i+1;
          cout<<c[i]<<" ";
        }
        cout<<" \n ";
        generaterandomarray(c);
        cout<<" \n ";

}
```

OUTPUT :



```
 Input the elements of array between 1 to 100 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 2
7 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65
66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
  The random input array generated ( 1 to 100 ) for program is
 84 87 32 7 94 66 2 93 14 22 63 39 72 10 75 49 41 4 73 44 28 40 33 12 58 62 34 15 23 24 6 78 13 59 77 29 65 46 21 90
 26 74 69 79 85 56 67 83 27 71 80 54 92 81 55 42 18 35 97 82 53 31 45 30 3 61 99 76 20 9 48 96 25 57 64 17 8 5 1 60
88 50 70 37 86 89 98 11 38 100 68 36 19 51 47 91 52 16 43 95
  Enter the option to choose 1) randselect or 2) select 1
 Implementation of the randselect program
  Enter the smallest k 4

  The smallest element k for randselect is
 4
```



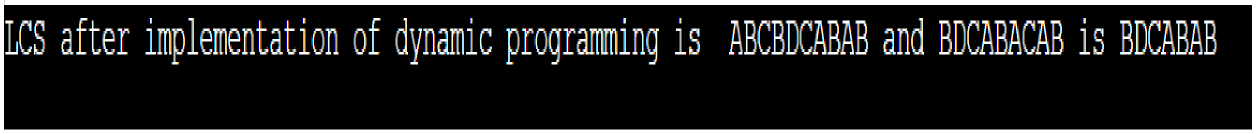
```
 Input the elements of array between 1 to 100 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 2
7 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65
66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
  The random input array generated ( 1 to 100 ) for program is
 84 87 32 7 94 66 2 93 14 22 63 39 72 10 75 49 41 4 73 44 28 40 33 12 58 62 34 15 23 24 6 78 13 59 77 29 65 46 21 90
 26 74 69 79 85 56 67 83 27 71 80 54 92 81 55 42 18 35 97 82 53 31 45 30 3 61 99 76 20 9 48 96 25 57 64 17 8 5 1 60
88 50 70 37 86 89 98 11 38 100 68 36 19 51 47 91 52 16 43 95
  Enter the option to choose 1) randselect or 2) select 2
Implementaion of the select program
 Enter the smallest k 3

 The smallest k elements for Select is 3
```

```
******************Algorithm for the Dynamic programming in lcs****************************
/* Dynamic Programming implementation of Longest Common Subsequence  problem */
#include<iostream>
#include<cstring>
#include<cstdlib>
using namespace std;

void lcs( char *word1, char *word2, int x, int y )
{
```

```
int C[x+1][y+1];

/* Using the Botton Up approach .*/
for (int i=0; i<=x; i++)
{
        for (int j=0; j<=y; j++)
        {
        if (i == 0 || j == 0)
                C[i][j] = 0;            // Setting the array at zero
        else if (word1[i-1] == word2[j-1])
                C[i][j] = C[i-1][j-1] + 1;
        else
                C[i][j] = max(C[i-1][j], C[i][j-1]);
        }
}

// Printing the  Longest Common Subsequence
int referance = C[x][y];

char lcs[referance+1];
lcs[referance] = '\0';

//Storing charactes in the lcs
int i = x, j = y;
while (i > 0 && j > 0)
{

        if (word1[i-1] == word2[j-1])
        {
                lcs[referance-1] = word1[i-1];
                i--; j--; referance--;
        }

        else if (C[i-1][j] > C[i][j-1])
                i--;
        else
                j--;
}

cout << "LCS after implementation of dynamic programming is  " << word1 << " and " << word2 << " is "
<< lcs;
}

int main()
```

```
{
char word1[] = "ABCBDCABAB";
char word2[] = "BDCABACAB";
int x = strlen(word1);
int y = strlen(word2);
lcs(word1, word2, x, y);
return 0;
}
```

OUTPUT :



```
LCS after implementation of dynamic programming is  ABCBDCABAB and BDCABACAB is BDCABAB
```