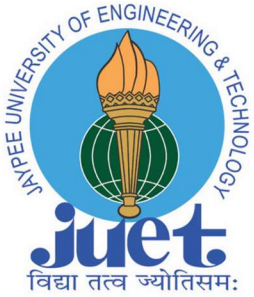
Jaypee University of Engineering and Technology, Guna



[LAB ACTIVITY 6]

DATA STRUCTURES LAB

18B17CI371

Name: Palash Mishra

Enroll No. : 201B172

Questions:

1. WAP to implement a function Rdm(n) which returns an array of random numbers{between 0 to 99}, where n is the size of array. (Hint: use dynamic memory allocation concept)

NOTE: Use Rdm function by putting it into separate header file for questions number 2 through 6.

Solution:

```
int *Rdm(int n)
{
  int iSecret, iGuess;
  int *p;
  p = new int(n);
  srand(time(NULL));
  for (int i = 0; i < n; i++)
  {
    p[i] = rand() % 100;
  }
  return p;
}
```

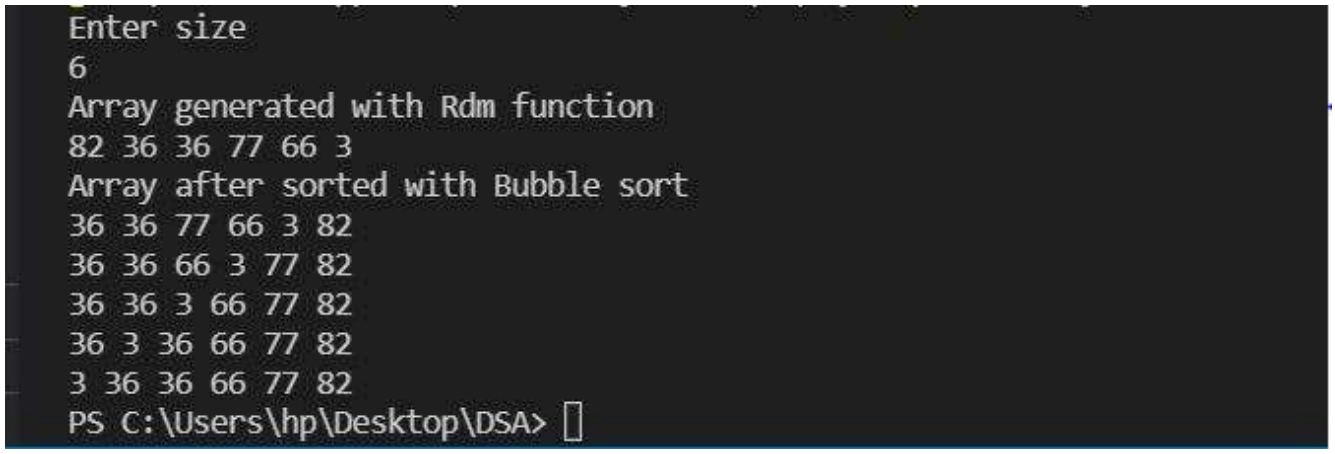
2. WAP to implement the bubble sort and show the output of each pass.

Solution:

```
#include<iostream>
#include <stdlib.h>
#include <time.h>
```

```cpp
#include"Rdm.h"
using namespace std;
void Bubble_sort(int arr[],int size)
{
    int i,j,temp;
    cout<<endl<<"Array after sorted with Bubble sort";
    for(int i=0;i<size-1;i++)
    {
        cout<<endl;
        for(int j=0;j<size-i-1;j++)
        {
            if(arr[j]>arr[j+1])
            {
                temp=arr[j];
                arr[j]=arr[j+1];
                arr[j+1]=temp;
            }
        }
        for(int z=0;z<size;z++)
        {
            cout<<arr[z]<<" ";
        }
    }
}
int main()
{
    int size;
    cout<<"Enter size"<<endl;
    cin>>size;
    int *z=Rdm(size);
    cout<<"Array generated with Rdm function"<<endl;
    for(int i=0;i<size;i++)
    {
        cout<<z[i]<<" ";
    }
    Bubble_sort(z,size);
    delete z;
    return 0;
}
```

Output:



3. WAP to implement the selection sort and show the output of each pass.

Solution:

```
#include<iostream>
#include <stdlib.h>
#include <time.h>
#include"Rdm.h"
using namespace std;
int Selection_sort(int arr[],int size)
{
    int i,j,min,temp;
    cout<<endl<<"Array after sorted with Selection sort";
    for(i=0;i<size-1;i++)
    {
        cout<<endl;
        min=i;
        for(j=i+1;j<size;j++)
        {
            if(arr[j]<arr[min])
            {
                min=j;
            }
        }
        temp=arr[min];
```

```
        arr[min]=arr[i];
        arr[i]=temp;
        for(int z=0;z<size;z++)
        {
            cout<<arr[z]<<" ";
        }
    }
}
int main()
{
    int size;
    cout<<"Enter size"<<endl;
    cin>>size;
    int *z=Rdm(size);
    cout<<"Array generated with Rdm function"<<endl;
    for(int i=0;i<size;i++)
    {
        cout<<z[i]<<" ";
    }
    Selection_sort(z,size);
    delete z;
    return 0;
}
```

Output:

```
Enter size
6
Array generated with Rdm function
17 89 5 32 19 16
Array after sorted with Selection sort
5 89 17 32 19 16
5 16 17 32 19 89
5 16 17 32 19 89
5 16 17 19 32 89
5 16 17 19 32 89
PS C:\Users\hp\Desktop\DSA>
```

4. WAP to implement the insertion sort and show the output of each pass.

Solution:

```
#include<iostream>
#include <stdlib.h>
#include <time.h>
#include"Rdm.h"
using namespace std;
int insertion_sort(int arr[],int size)
{
    int key,ind,i;
    cout<<endl<<"Array after sorted with Insertion sort";
    for(i=1;i<size;i++)
    {
        cout<<endl;
        key=arr[i];
        ind=i;
        while(arr[ind-1]>key && ind>=1)
        {
            arr[ind]=arr[ind-1];
            ind--;
        }
        arr[ind]=key;
    for(int z=0;z<size;z++)
    {
        cout<<arr[z]<<" ";
    }
    }
}
int main()
{
    int size;
    cout<<"Enter size"<<endl;
    cin>>size;
    int *z=Rdm(size);
    cout<<"Array generated with Rdm function"<<endl;
    for(int i=0;i<size;i++)
```

```
    {
        cout<<z[i]<<" ";
    }
    insertion_sort(z,size);
    delete z;
    return 0;
}
```

Output:

```
Enter size
6
Array generated with Rdm function
84 2 73 77 75 71
Array after sorted with Insertion sort
2 84 73 77 75 71
2 73 84 77 75 71
2 73 77 84 75 71
2 73 75 77 84 71
2 71 73 75 77 84
```

5. WAP to implement the quick sort and show the output of each pass.

Solution:

Output:

6. WAP to implement the merge sort and show the output of each pass.

Advanced Problems

Solution:

Output:

7. WAP to sort a character array using insertion sort in alphabetic order and print number of shifts.

Solution:

```
#include <iostream>
#include<ctype.h>
#include<string.h>
using namespace std;
void InsertionSort(char *p, int size)
{
    int i, j, key;
    for (i = 1; i < size; i++)
    {
        key = p[i];
        j = i;
        while (p[j - 1] < key && j >= 1)
        {
            p[j] = p[j - 1];
            j--;
        }
        p[j] = key;
    }
}
int main()
{
    int len;
    char str[]="JAYPEE UNIVERSITY OF ENGINEERING AND
TECHNOLOGY";
    len=strlen(str);
    InsertionSort(str, len);
    for (int i = 0; i < len; i++)
    {
        cout << str[i] << " ";
    }
    return 0;
}
```

Output:

```
PS C:\Users\hp\Desktop\DSA> cd "c:\Users\hp\Desktop\DSA\" ; if ($?) { g++ question7.cpp
Y Y Y V U T T S R R P O O O N N N N N N L J I I I I H G G G F E E E E E E E D C A A
PS C:\Users\hp\Desktop\DSA>
```

8. WAP to partition the array in two parts by taking the position of last element as pivot element and leave first sub array with smaller value elements and second sub array with greater value elements.

Solution:

```
#include <stdio.h>
int partition(int arr[], int low, int high)
{
    int temp;
    int pivot = arr[high];
    int i = (low - 1);
    for (int j = low; j <= high - 1; j++)
    {
        if (arr[j] <= pivot)
        {
            i++;
            temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
    temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;
    return (i + 1);
}
```

```c
void quick_sort(int arr[], int low, int high)
{
    if (low < high)
    {
        int pi = partition(arr, low, high);
        quick_sort(arr, low, pi - 1);
        quick_sort(arr, pi + 1, high);
    }
}

int print(int arr[], int n)
{
    printf("\n Left Half \n");
    for (int i = 0; i < n / 2; i++)
    {
        printf("%d ", arr[i]);
    }
}
int print1(int arr[], int n)
{
    printf("\n Right Half \n");
    for (int i = n / 2; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
}

int main()
{
    int n, i;
    scanf("%d", &n);
    int arr[n];
    for (i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }
    quick_sort(arr, 0, n - 1);
    print(arr, n);
    print1(arr, n);
}
```

Output:

```
5
14 2 91 36 6

 Left Half
2 6
 Right Half
14 36 91
```

9. WAP to insert an element in sorted array and after insertion order should not change.

Sample input : 2, 4, 5, 6, 8, 9, 10, 13, 15 and 7

Sample output : 2, 4, 5, 6, 7, 8, 9, 10, 13, 15

Solution:

```
#include <iostream>
using namespace std;
int insert(int a[], int numOfElements, int size, int num)
{
    if (numOfElements == size)
        return numOfElements;

    int p;
    for (int i = 0; i < numOfElements; i++)
        if (num < a[i])
        {
            p = i;
            break;
        }

    for (int i = numOfElements; i >= p; i--)
        a[i] = a[i - 1];

    a[p] = num;

    numOfElements++;
```

```
    return numOfElements;
}
int main()
{
    int arr[14] = {2, 3, 5, 5, 6, 14, 15, 19, 20, 25, 63,
54};
    int size = sizeof(arr) / sizeof(arr[0]);
    int num_of_elements = 12;
    int num;
    cout << "Enter the number you wanna add : ";
    cin >> num;
    cout << "Before Insertion" << endl;
    for (int i = 0; i < num_of_elements; i++)
    {
        cout << arr[i] << " ";
    }
    cout<<endl;
    num_of_elements = insert(arr, num_of_elements, size,
num);
    for (int i = 0; i < num_of_elements; i++)
    {
        cout << arr[i] << " ";
    }
    return 0;
}
```

Output:

```
Enter the number you wanna add : 9
Before Insertion
2 3 5 5 6 14 15 19 20 25 63 54
2 3 5 5 6 9 14 15 19 20 25 63 54
```