NAME-

SHIKSHA

SHRIVASTAVA

ROLL-2022338

EXAMINATION ROLL-

2206670031

SEM-IV

SUB- DAA

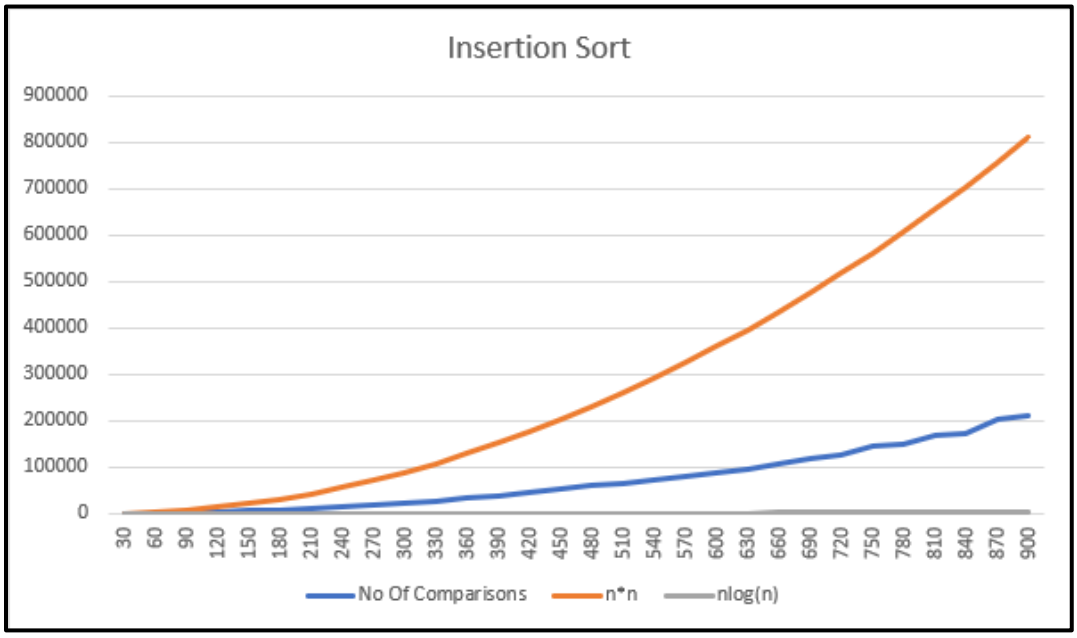
1. Write a program to sort the elements of an array using Insertion sort.

CODE:

```
#include<iostream>
#include<cstdlib>
#include<fstream>
using namespace std;
template <class t>
class insort
{
   public:
   t a[700];
   int n;
   void input(int m);
   void output();
   int isort(int m);
};
template <class t>
void insort<t>::input(int m)
{
   n=m;
   for(int i=1;i<=n;i++)
   {
      a[i]=rand()%100;
      cout<<a[i]<<",";
   }
}
template <class t>
void insort<t>::output()
{
   for(int i=1;i<=n;i++)

          {
      cout<<a[i]<<",";
   }
}
template <class t>
int insort<t>::isort(int m)
{
   int i,j,key;
```

```
    int count=0;
    for(j=2;j<=n;j++)
    {
       key=a[j];
       i=j-1;
       while(i>0 && a[i]>key)
       {
         a[i+1]=a[i];
         i=i-1;
         count++;

             }
             count++;
    a[i+1]=key;
             }
    cout<<"\nSorted array: \t";
    output();
    cout<<"\nThe number of comparisons are : "<<count<<endl;
             return count;
}
int main()
{
    int k=1,m=30,c;
    insort<int> srt;
    ofstream fout;
    fout.open("insertion_sort1.csv");
    while(k<=20)
    {
       srt.input(m);
       c=srt.isort(m);
                        k++;
                        m=m+30;
       if(fout)
       {
          fout<<srt.n<<","<<c;
          fout<<endl;
       }
       srt.output();
    }
             fout.close();
    return 0;
}
```

OUTPUT:



2.Write a program to sort the elements of an array using merge sort.

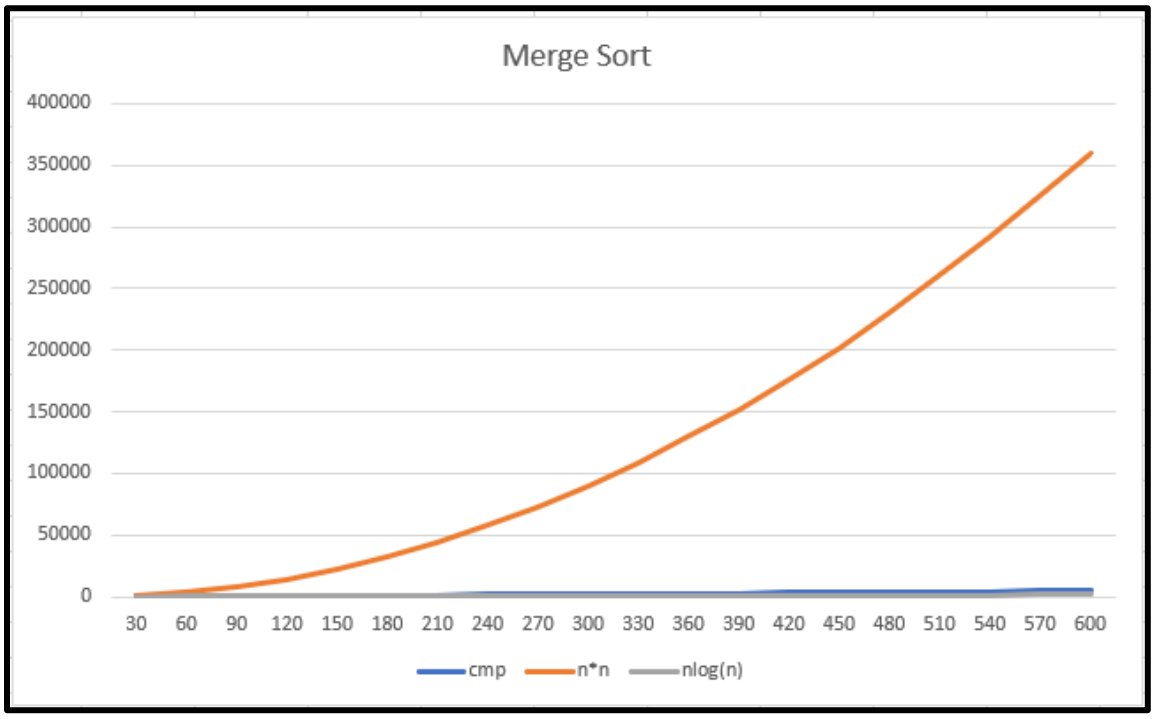
CODE:

```
#include<iostream>
#include<fstream>
using namespace std;
template<class t>
class MergeSort{
t a[1000];
int count;
public:
void input(int n);
int msort( int p,int r);
void merge(int p, int q, int r);
void display(int n);
};
template<class t>
void MergeSort<t>::input(int n){
for (int i=0;i<n;i++){
a[i]=rand()%1000;
```

```
}
}
template<class t>
int MergeSort<t>::msort(int p, int r){
count=0;
if (p<r){
int q=(p+r)/2;
count+=msort(p,q);
count+=msort(q+1,r);
merge(p,q,r);
}
return count;
}
template<class t>
void MergeSort<t>::merge(int p,int q,int r){
int nL=q-p+1;
int nR=r-q;
t *L=new t[nL];
t *R=new t[nR];
for (int i=0;i<nL;i++){
L[i]=a[p+i];
}
for (int j=0;j<nR;j++){
R[j]=a[q+j+1];
}
int i=0,j=0,k=p;
while (i<nL && j<nR){
if (L[i]<=R[j]){
a[k++]=L[i++];
count++;
}else{
a[k++]=R[j++];
count++;
}
}
while(i<nL){
a[k++]=L[i++];
}
while(j<nR){
a[k++]=R[j++];
}
delete[] L;
```

```
delete[] R;
}
template<class t>
void MergeSort<t>::display(int n){
for (int i=0;i<n;i++){
cout<<a[i]<<" ";
}
cout<<endl;
}
int main(){
MergeSort<int> obj;
ofstream fout;
int a;
fout.open("msort_data.csv");
int k=1,n=30;
while (k<=20){
cout<<k<<" iteration";
cout<<"\nUnsorted"<<endl;
obj.input(n);
obj.display(n);
a=obj.msort(0,n-1);
fout<<n<<','<<a<<'\n';
cout<<"\nSorted"<<endl;
obj.display(n);
n+=30;
k++;
cout<<"\n";
}
fout.close();
return 0;
}
```

OUTPUT:



3. Write a program to sort the elements of an array using Heap Sort.

CODE:

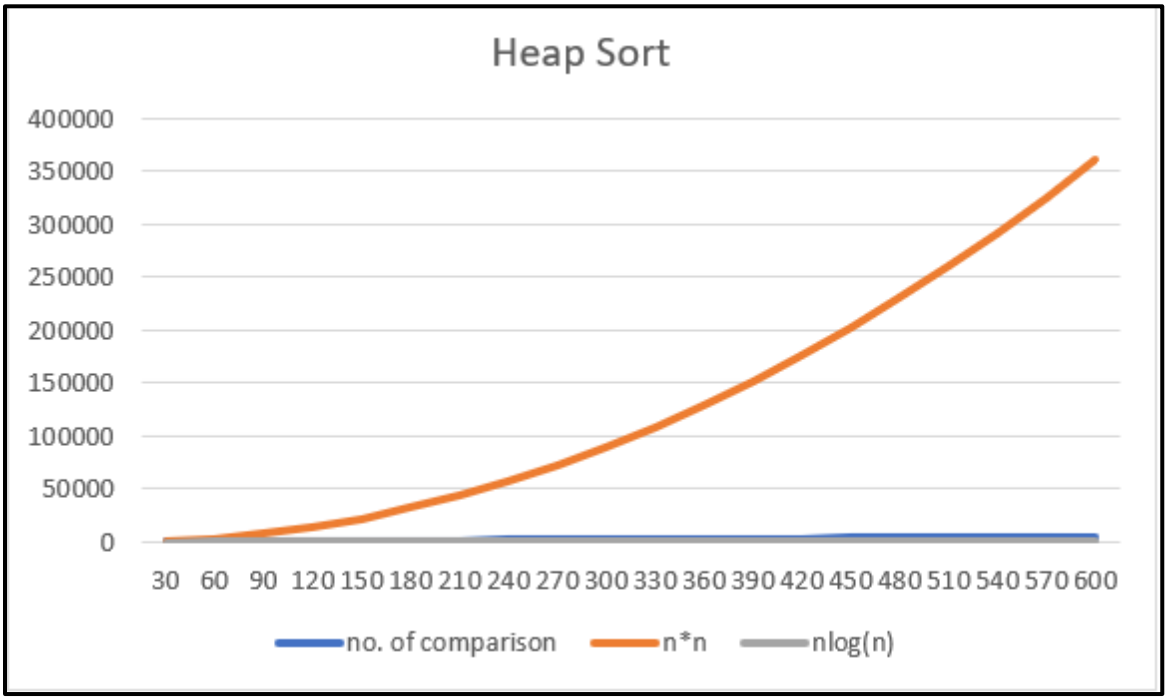
```
#include<iostream>
#include<fstream>
using namespace std;

template<class t>
class HeapSort{
   t *a;
   int heap_size;
   //int count;
public:
   int count;
   HeapSort();
   ~HeapSort();
   void input(int n);
   void heapsort(int n);
   void build_max_heap(int n);
```

```
    void max_heapify(int i, int n);
    void display(int n);
};

template<class t>
HeapSort<t>::HeapSort() {
    a = 0;
    heap_size = 0;
    count = 0;
}

template<class t>
HeapSort<t>::~HeapSort() {
    delete[] a;
}

template<class t>
void HeapSort<t>::input(int n){
    heap_size = n;
    a = new t[n];
    for (int i = 0; i < n; i++) {
        a[i] = rand() % 1000;
    }
}

template<class t>
void HeapSort<t>::heapsort(int n) {
    count=0;
    build_max_heap(n);
    for (int i = n - 1; i > 0; i--) {
        swap(a[0], a[i]);
        heap_size--;
        max_heapify(0, heap_size);
    }
}

template<class t>
void HeapSort<t>::build_max_heap(int n) {
    for (int i = n / 2 - 1; i >= 0; i--) {
```

```
        max_heapify(i, n);
    }
}

template<class t>
void HeapSort<t>::max_heapify(int i, int n) {
    int largest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;

    if (left < n && a[left] > a[largest]) {
        largest = left;
    }
    if (right < n && a[right] > a[largest]) {
        largest = right;
    }

    if (largest != i) {
        swap(a[i], a[largest]);
        max_heapify(largest, n);
    }
    count++;
}

template<class t>
void HeapSort<t>::display(int n){
    for (int i = 0; i < n; i++) {
        cout << a[i] << " ";
    }
    cout << endl;
}

int main(){
    HeapSort<int> obj;
    ofstream fout;
    int a;
    fout.open("heapsort_data.csv");
    int k = 1, n = 30;
    while (k <= 20) {
```

```
            cout << k << " iteration";
            cout << "\nUnsorted" << endl;
            obj.input(n);
            obj.display(n);
            obj.heapsort(n);
            fout << n << ',' << obj.count << '\n';
            cout << "\nSorted" << endl;
            obj.display(n);
            n += 30;
            k++;
            cout << "\n";
        }
        fout.close();
        return 0;
    }
```

OUTPUT:



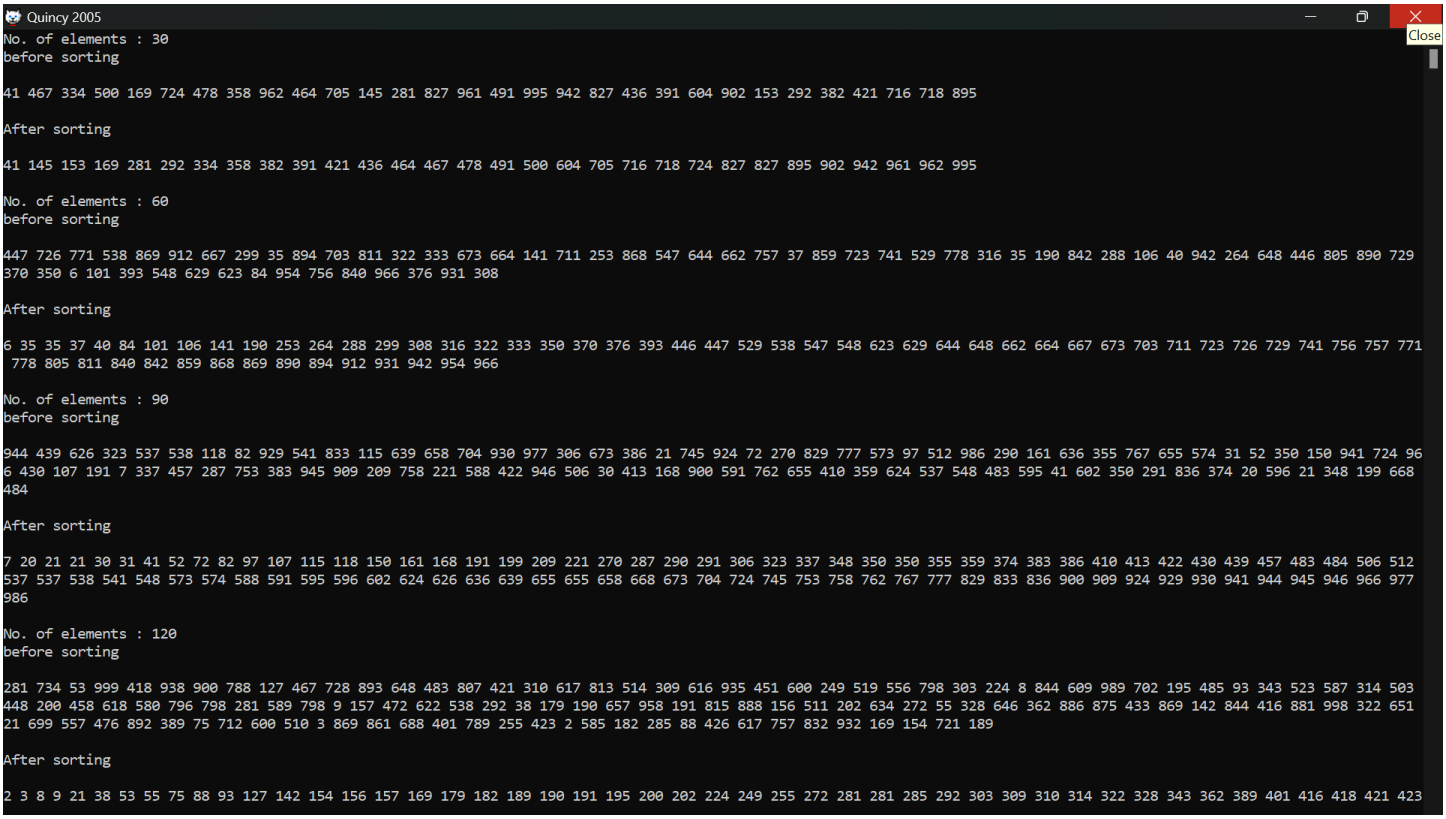
4.Write a program to sort the elements of an array using Quick Sort

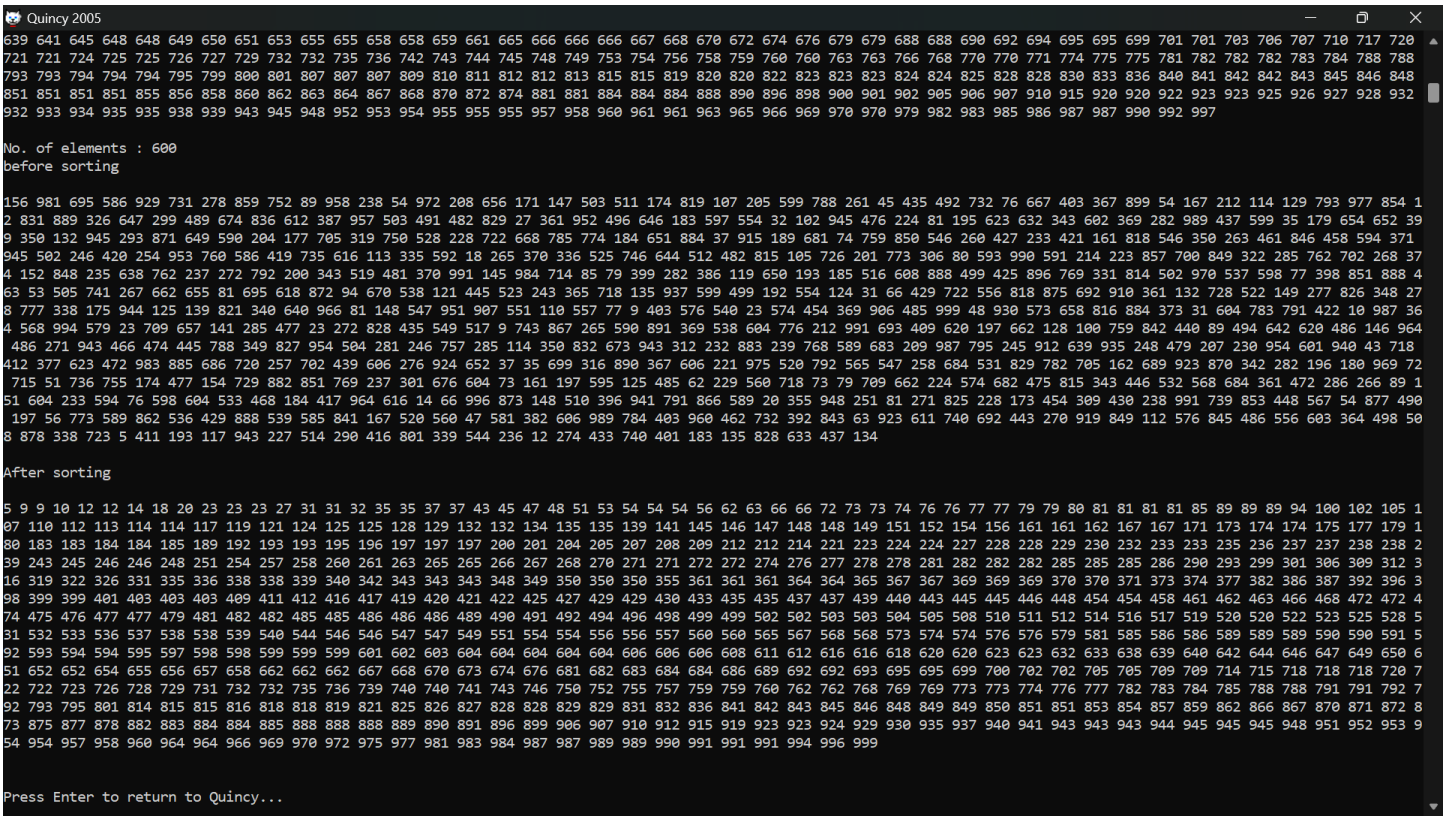
CODE:

```
#include<iostream>
#include<fstream>
using namespace std;
template <class t>
class quicksort
{
public:
t a[1000];
int num;

quicksort(int n)
{
num = n;
}
void input(int n);
void qsort(int a[], int p, int r);
int partition(int a[], int p, int r);
void output(int n);
};
template <class t>
void quicksort <t>:: input(int n)
{
for (int i = 1; i <= n; i++)
{
a[i] = rand() % 1000;
}
}
template <class t>
int quicksort <t>:: partition(int a[], int p, int r)
{
t x = a[r];
int j;
int i = p - 1;
for (j = p;j <= r - 1;j++)
{
if (a[j] <= x)
{
i++;
swap(a[i], a[j]);
}
}
swap(a[i+1], a[r]);
return i + 1;
}
template <class t>
void quicksort <t>:: qsort(int a[], int p, int r)
{
```

```
int q;
if (p < r)
{
q = partition(a, p, r);
qsort(a, p, q - 1);
qsort(a, q + 1, r);
}
}
template <class t>
void quicksort <t>:: output(int n)
{
for (int i = 1; i <= n; i++)
{
cout << a[i] << " ";
}
cout << endl << endl;
}
int main()
{
int k = 1, n = 30;
ofstream fout;
fout.open("qsort_data.csv");
quicksort<int> ob(n);
while (k <= 20)
{
ob.input(n);
cout << "No. of elements : " << n << endl;
cout << "before sorting " << endl << endl;
//int a[1000];
ob.output(n);
ob.qsort(ob.a,1, n);
cout << "After sorting" << endl << endl;;
ob.output(n);
//    fout<<n<<','<<c<<'\n';
n += 30;
k++;
}
fout.close();
return 0;
}
```

OUTPUT:


Quincy 2005
Close
No. of elements : 30
before sorting
41 467 334 500 169 724 478 358 962 464 705 145 281 827 961 491 995 942 827 436 391 604 902 153 292 382 421 716 718 895
After sorting
41 145 153 169 281 292 334 358 382 391 421 436 464 467 478 491 500 604 705 716 718 724 827 827 895 902 942 961 962 995
No. of elements : 60
before sorting
447 726 771 538 869 912 667 299 35 894 703 811 322 333 673 664 141 711 253 868 547 644 662 757 37 859 723 741 529 778 316 35 190 842 288 106 40 942 264 648 446 805 890 729
370 350 6 101 393 548 629 623 84 954 756 840 966 376 931 308
After sorting
6 35 35 37 40 84 101 106 141 190 253 264 288 299 308 316 322 333 350 370 376 393 446 447 529 538 547 548 623 629 644 648 662 664 667 673 703 711 723 726 729 741 756 757 771
778 805 811 840 842 859 868 869 890 894 912 931 942 954 966
No. of elements : 90
before sorting
944 439 626 323 537 538 118 82 929 541 833 115 639 658 704 930 977 306 673 386 21 745 924 72 270 829 777 573 97 512 986 290 161 636 355 767 655 574 31 52 350 150 941 724 96
6 430 107 191 7 337 457 287 753 383 945 909 209 758 221 588 422 946 506 30 413 168 900 591 762 655 410 359 624 537 548 483 595 41 602 350 291 836 374 20 596 21 348 199 668
484
After sorting
7 20 21 21 30 31 41 52 72 82 97 107 115 118 150 161 168 191 199 209 221 270 287 290 291 306 323 337 348 350 350 355 359 374 383 386 410 413 422 430 439 457 483 484 506 512
537 537 538 541 548 573 574 588 591 595 596 602 624 626 636 639 655 655 658 668 673 704 724 745 753 758 762 767 777 829 833 836 900 909 924 929 930 941 944 945 946 966 977
986
No. of elements : 120
before sorting
281 734 53 999 418 938 900 788 127 467 728 893 648 483 807 421 310 617 813 514 309 616 935 451 600 249 519 556 798 303 224 8 844 609 989 702 195 485 93 343 523 587 314 503
448 200 458 618 580 796 798 281 589 798 9 157 472 622 538 292 38 179 190 657 958 191 815 888 156 511 202 634 272 55 328 646 362 886 875 433 869 142 844 416 881 998 322 651
21 699 557 476 892 389 75 712 600 510 3 869 861 688 401 789 255 423 2 585 182 285 88 426 617 757 832 932 169 154 721 189
After sorting
2 3 8 9 21 38 53 55 75 88 93 127 142 154 156 157 169 179 182 189 190 191 195 200 202 224 249 255 272 281 281 285 292 303 309 310 314 322 328 343 362 389 401 416 418 421 423


Quincy 2005
639 641 645 648 648 649 650 651 653 655 655 658 658 659 661 665 666 666 666 667 668 670 672 674 676 679 679 688 688 690 692 694 695 695 699 701 701 703 706 707 710 717 720
721 721 724 725 725 726 727 729 732 732 735 736 742 743 744 745 748 749 753 754 756 758 759 760 760 763 763 766 768 770 770 771 774 775 775 781 782 782 782 783 784 788 788
793 793 794 794 794 795 799 800 801 807 807 807 809 810 811 812 812 813 815 815 819 820 820 822 823 823 823 824 824 825 828 828 830 833 836 840 841 842 842 843 845 846 848
851 851 851 851 855 856 858 860 862 863 864 867 868 870 872 874 881 881 884 884 884 888 890 896 898 900 901 902 905 906 907 910 915 920 920 922 923 923 925 926 927 928 932
932 933 934 935 935 938 939 943 945 948 952 953 954 955 955 955 957 958 960 961 961 963 965 966 969 970 970 979 982 983 985 986 987 987 990 992 997
No. of elements : 600
before sorting
156 981 695 586 929 731 278 859 752 89 958 238 54 972 208 656 171 147 503 511 174 819 107 205 599 788 261 45 435 492 732 76 667 403 367 899 54 167 212 114 129 793 977 854 1
2 831 889 326 647 299 489 674 836 612 387 957 503 491 482 829 27 361 952 496 646 183 597 554 32 102 945 476 224 81 195 623 632 343 602 369 282 989 437 599 35 179 654 652 39
9 350 132 945 293 871 649 590 204 177 705 319 750 528 228 722 668 785 774 184 651 884 37 915 189 681 74 759 850 546 260 427 233 421 161 818 546 350 263 461 846 458 594 371
945 502 246 420 254 953 760 586 419 735 616 113 335 592 18 265 370 336 525 746 644 512 482 815 105 726 201 773 306 80 593 990 591 214 223 857 700 849 322 285 762 702 268 37
4 152 848 235 638 762 237 272 792 200 343 519 481 370 991 145 984 714 85 79 399 282 386 119 650 193 185 516 608 888 499 425 896 769 331 814 502 970 537 598 77 398 851 888 4
63 53 505 741 267 662 655 81 695 618 872 94 670 538 121 445 523 243 365 718 135 937 599 499 192 554 124 31 66 429 722 556 818 875 692 910 361 132 728 522 149 277 826 348 27
8 777 338 175 944 125 139 821 340 640 966 81 148 547 951 907 551 110 557 77 9 403 576 540 23 574 454 369 906 485 999 48 930 573 658 816 884 373 31 604 783 791 422 10 987 36
4 568 994 579 23 709 657 141 285 477 23 272 828 435 549 517 9 743 867 265 590 891 369 538 604 776 212 991 693 409 620 197 662 128 100 759 842 440 89 494 642 620 486 146 964
486 271 943 466 474 445 788 349 827 954 504 281 246 757 285 114 350 832 673 943 312 232 883 239 768 589 683 209 987 795 245 912 639 935 248 479 207 230 954 601 940 43 718
412 377 623 472 983 885 686 720 257 702 439 606 276 924 652 37 35 699 316 890 367 606 221 975 520 792 565 547 258 684 531 829 782 705 162 689 923 870 342 282 196 180 969 72
715 51 736 755 174 477 154 729 882 851 769 237 301 676 604 73 161 197 595 125 485 62 229 560 718 73 79 709 662 224 574 682 475 815 343 446 532 568 684 361 472 286 266 89 1
51 604 233 594 76 598 604 533 468 184 417 964 616 14 66 996 873 148 510 396 941 791 866 589 20 355 948 251 81 271 825 228 173 454 309 430 238 991 739 853 448 567 54 877 490
197 56 773 589 862 536 429 888 539 585 841 167 520 560 47 581 382 606 989 784 403 960 462 732 392 843 63 923 611 740 692 443 270 919 849 112 576 845 486 556 603 364 498 50
8 878 338 723 5 411 193 117 943 227 514 290 416 801 339 544 236 12 274 433 740 401 183 135 828 633 437 134
After sorting
5 9 9 10 12 12 14 18 20 23 23 23 27 31 31 32 35 35 37 37 43 45 47 48 51 53 54 54 54 56 62 63 66 66 72 73 73 74 76 76 77 77 79 79 80 81 81 81 81 85 89 89 89 94 100 102 105 1
07 110 112 113 114 114 117 119 121 124 125 125 128 129 132 132 134 135 135 139 141 145 146 147 148 148 149 151 152 154 156 161 161 162 167 167 171 173 174 174 175 177 179 1
80 183 183 184 184 185 189 192 193 193 195 196 197 197 197 200 201 204 205 207 208 209 212 212 214 221 223 224 224 227 228 228 229 230 232 233 233 235 236 237 237 238 238 2
39 243 245 246 246 248 251 254 257 258 260 261 263 265 265 266 267 268 270 271 271 272 272 274 276 277 278 278 281 282 282 282 285 285 285 286 290 293 299 301 306 309 312 3
16 319 322 326 331 335 336 338 338 339 340 342 343 343 343 348 349 350 350 350 355 361 361 361 364 364 365 367 367 369 369 369 370 370 371 373 374 377 382 386 387 392 396 3
98 399 399 401 403 403 403 409 411 412 416 417 419 420 421 422 425 427 429 429 430 433 435 435 437 437 439 440 443 445 445 446 448 454 454 458 461 462 463 466 468 472 472 4
74 475 476 477 477 479 481 482 482 485 485 486 486 486 489 490 491 492 494 496 498 499 499 502 502 503 503 504 505 508 510 511 512 514 516 517 519 520 520 522 523 525 528 5
31 532 533 536 537 538 538 539 540 544 546 546 547 547 549 551 554 554 556 556 557 560 560 565 567 568 568 573 574 574 576 576 579 581 585 586 586 589 589 589 590 590 591 5
92 593 594 594 595 597 598 598 599 599 599 601 602 603 604 604 604 604 604 606 606 606 608 611 612 616 616 618 620 620 623 623 632 633 638 639 640 642 644 646 647 649 650 6
51 652 652 654 655 656 657 658 662 662 662 667 668 670 673 674 676 681 682 683 684 684 686 689 692 692 693 695 695 699 700 702 702 705 705 709 709 714 715 718 718 718 720 7
22 722 723 726 728 729 731 732 732 735 736 739 740 740 741 743 746 750 752 755 757 759 759 760 762 762 768 769 769 773 773 774 776 777 782 783 784 785 788 788 791 791 792 7
92 793 795 801 814 815 815 816 818 818 819 821 825 826 827 828 828 829 829 831 832 836 841 842 843 845 846 848 849 849 850 851 851 853 854 857 859 862 866 867 870 871 872 8
73 875 877 878 882 883 884 884 885 888 888 888 889 890 891 896 899 906 907 910 912 915 919 923 923 924 929 930 935 937 940 941 943 943 943 944 945 945 945 948 951 952 953 9
54 954 957 958 960 964 964 966 969 970 972 975 977 981 983 984 987 987 989 989 990 991 991 991 994 996 999
Press Enter to return to Quincy...

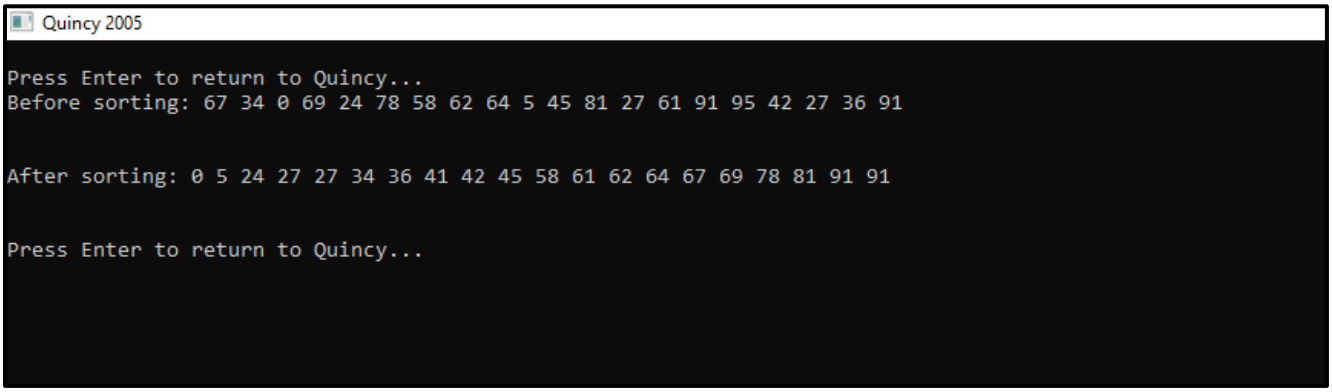
6.Write a program to sort the elements of an array using Count Sort

CODE:

```
#include<iostream>
using namespace std;

template <class t>
class CountingSort{
	public:
		t A[31],B[31];
		int num;
		CountingSort()
		{
			num=31;
		}
	void input(int num);
	int csort(int A[],int num);
	void display(int num);
};

template <class t>
void CountingSort<t>::input(int num){
	for (int i=1;i<=num;i++)
	{
		A[i]=rand()%100;
		cout<<A[i]<<" ";
	}
}

template <class t>
int CountingSort<t>::csort(int A[],int num)
{
	int i,j;
	int C[101];
	for (i=0;i<=100;i++)
		C[i]=0;
	for (j=1;j<=num-1;j++)
		C[A[j]]=C[A[j]]+1;
	for (i=1;i<=100;i++)
		C[i]=C[i]+C[i-1];
	for (j=num-1;j>=1;j--)
	{
		B[C[A[j]]]=A[j];
		C[A[j]]=C[A[j]]-1;
```

```
        }
//          for (int i=1;i<n;i++)
//                  cout<<A[i]<<" ";

}

template <class t>
void CountingSort<t>::display(int num){
        for (int i=1;i<num;i++){
                cout<<B[i]<<" ";
        }
        cout<<endl;
}

int main()
{
//          int n=30;
        CountingSort<int> obj;


        cout<<"\nUnsorted"<<endl;
        obj.input(obj.num);
//          obj.display(obj.num);
        cout<<"\nSorted"<<endl;
        obj.csort(obj.A,obj.num);
        obj.display(obj.num);
        cout<<"\n";
        return 0;
}
```

OUTPUT:



7.Display the data stored in a given graph using the Breadth first search algorithm.

CODE:

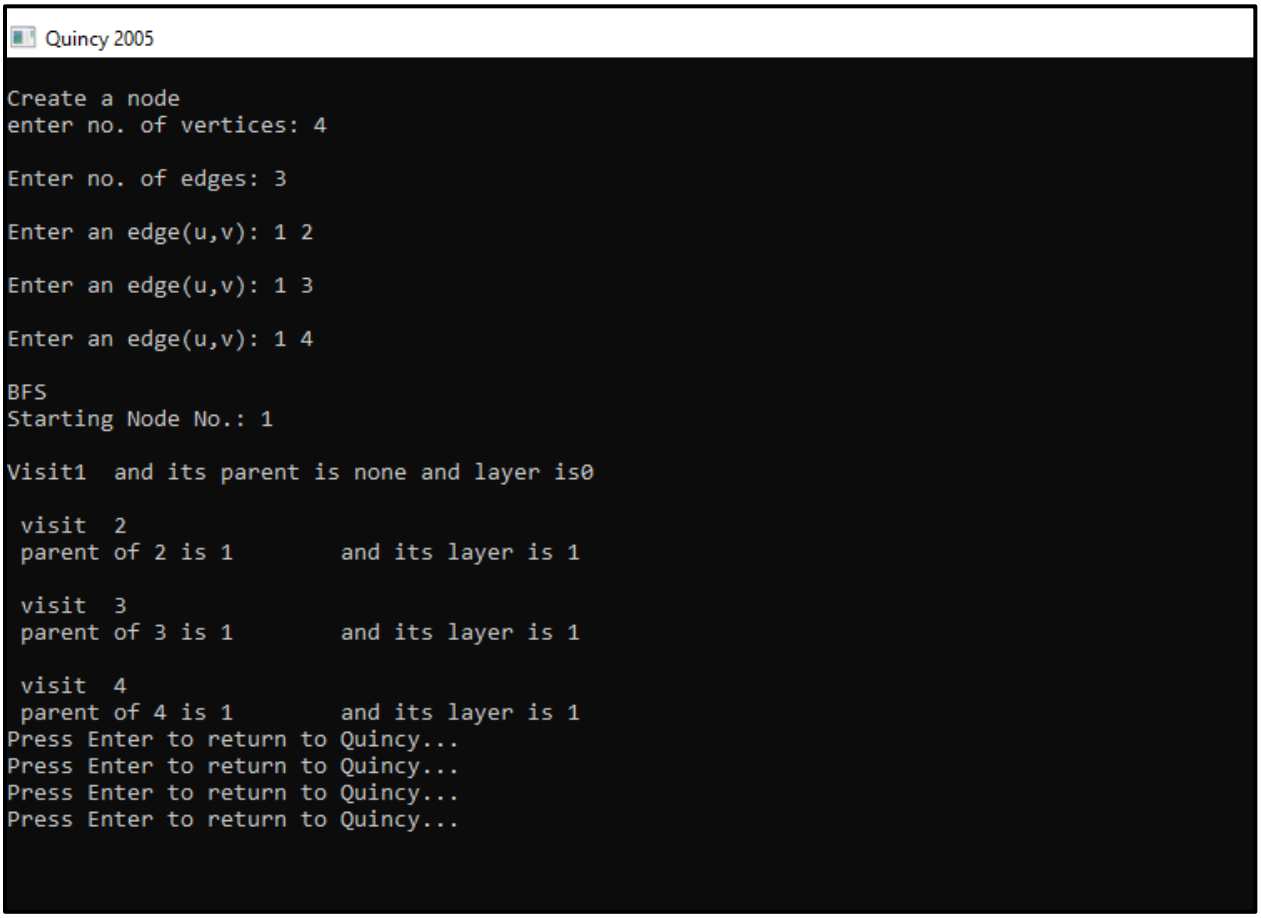
```
#include<iostream>
#include<stdlib.h>
using namespace std;
#define MAX 20

typedef struct Q
{
        int data[MAX];
        int R, F;
}Q;

typedef struct node
{
        struct node*next;
        int vertex;
}node;

void enqueue(Q *, int);
int dequeue(Q *);
int empty(Q*);
int full(Q*);
void BFS(int);
void readgraph();
void insert(int vi, int vj);
int discovered[MAX];
int layer[MAX], parent[MAX];
node *G[20];
int n;

int main()
{
        int i;
                cout<<"\nCreate a node";
        readgraph();
                cout<<"\nBFS";
                cout<<"\nStarting Node No.: ";
        cin>>i;
        BFS(i);
        return 0;
}

void BFS(int v)
{
```

```
int w,i;
Q q;

node *p;
q.R=q.F=-1;
for(i=0;i<=n;i++)
discovered[i]=0;
enqueue(&q,v);
int ly=0;
layer[v]=ly;
parent[v]=-1;
cout<<"\nVisit"<<v;
cout<<"\tand its parent is none and layer is"<<layer[v];
discovered[v]=1;
ly=1;
while(!empty(&q))
 {
        v=dequeue(&q);

        for(p=G[v];p!=NULL;p=p->next)
        {
                w=p->vertex;
                if(discovered[w]==0)
                {
                        parent[w]=v;
                        layer[w]=layer[parent[w]]+1;
                        enqueue(&q,w);
                        discovered[w]=1;
                        cout<<"\n \n visit \t"<<w;
                        cout<<"\n parent of "<<w<<" is "<<parent[w];
                        cout<<"\t and its layer is "<<layer[w];
                }
        }
 }
}

int empty(Q*P)
{
        if(P->R==-1)
        return (1);
        return (0);
}

int full(Q*P)
{
if(P->R==MAX-1)
return(1);
return(0);
}

void enqueue(Q *P,int x)
{
        if(P->R==-1)
```

```
        {
                P->R=P->R+1;
                P->data[P->R]=x;
        }
}

int dequeue(Q *P)
{
        int x;
        x=P->data[P->F];
        if(P->R==P->F)
        {
                P->R=-1;
                P->F=-1;
        }
        else
        P->F=P->F+1;
        return(x);
}

void readgraph()
{
        int i,vi,vj,no_of_edges;
        cout<<"\nenter no. of vertices: ";
        cin>>n;

        for(i=0;i<n;i++)
                G[i]=NULL;
        cout<<"\nEnter no. of edges: ";
        cin>>no_of_edges;
        for(i=0;i<no_of_edges;i++)
        {
                cout<<"\nEnter an edge(u,v): ";
                cin>>vi>>vj;
                insert(vi,vj);
                insert(vj,vi);
        }
}

void insert(int vi,int vj)
{
        node *p,*q;

        q=new node;
        q->vertex=vj;
        q->next=NULL;
        if(G[vi]==NULL)
                G[vi]=q;
        else
        {
                p=G[vi];
                while(p->next!=NULL)
                        p=p->next;
```

```
                    p->next=q;
                }
        }
```

OUTPUT:



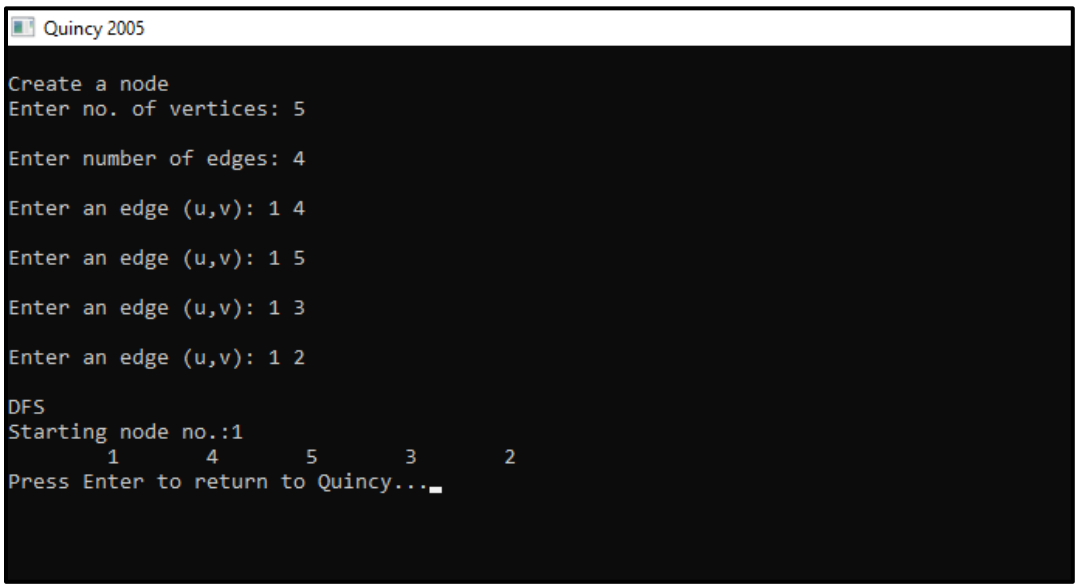
8.Display the data stored in a given graph using the Depth first search algorithm.

CODE:

```
#include <iostream>
using namespace std;
#define max 20
typedef struct Q
{
int data[max];
int R,F;
}Q;
typedef struct node
{
struct node* next;
int vertex;
}node;
void dfs(int);
void readgraph();
void insert(int vi, int vj);
int visited[max];
node *G[20];
int n;

int main()
{
int i;
cout<<"\nCreate a node";
readgraph();
cout<<"\nDFS";
cout<<"\nStarting node no.:";
cin>>i;
dfs(i);
return 0;
}

void dfs(int i)
{
node *p;
cout<<"\t"<<i;
p=G[i];
visited[i]=1;
while(p!=NULL)
{
i=p->vertex;
if(!visited[i])
dfs(i);
p=p->next;
}
}
```

```
void readgraph()
{
int i,vi,vj,no_of_edges;
cout<<"\nEnter no. of vertices: ";
cin>>n;
for (i=0;i<n;i++)
G[i]=NULL;
cout<<"\nEnter number of edges: ";
cin>>no_of_edges;
for (i=0;i<no_of_edges;i++)
{
cout<<"\nEnter an edge (u,v): ";
cin>>vi>>vj;
insert(vi,vj);
insert(vj,vi);
}
}
void insert(int vi, int vj)
{
node *p,*q;
q=new node;
q->vertex=vj;
q->next=NULL;
if(G[vi]==NULL)
G[vi]=q;
else
{
p=G[vi];
while(p->next!=NULL)
p=p->next;
p->next=q;
}
}
```

OUTPUT:



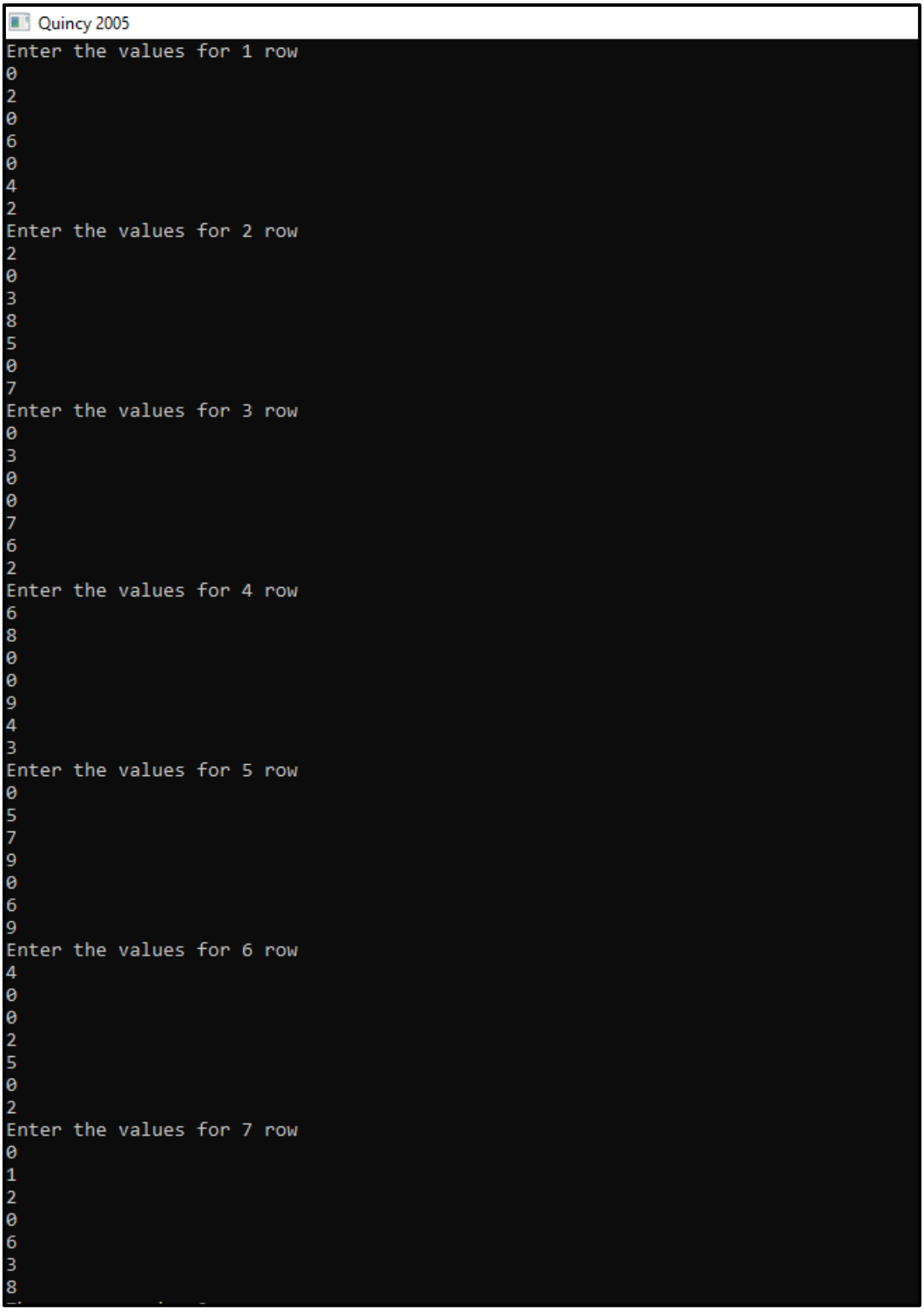
9.Write a program to determine a minimum spanning tree of a graph using the Prim's algorithm.

CODE:

```
#include <iostream>
using namespace std;

struct node
{
int fr,to,cost;
}p[8];

int c=0;
int temp1=0;
int temp=0;
void prims(int *a,int b[][7],int i,int j)
{
a[i]=1;
	while(c<6)
	{
		int min=9999;
		for(i=0;i<7;i++)
		{
			if(a[i]==1)
			{
				for(int j=0;j<7;)
				{
					if(b[i][j]>=min || b[i][j] ==0)
					j++;
					else if(b[i][j]<min)
					{
						min=b[i][j];
						temp=i;
						temp1=j;
					}
				}
			}
		}
		a[temp1]=1;
		p[c].fr=temp;
		p[c].to=temp1;
		p[c].cost=min;
		c++;
		b[temp][temp1]=b[temp1][temp]=1000;
	}
	for(int k=0;k<6;k++)
	{
		cout<<"The source node: "<<p[k].fr<<endl;
		cout<<"The Destination node: "<<p[k].to<<endl;
```

```
                cout<<"The weight of node: "<<p[k].cost<<endl;
        }
}

int main()
{

        int a[9];
        for(int i=0;i<7;i++)
        {
                a[i]=0;
        }
        int b[7][7];
        for(int i=0;i<7;i++)
        {
        cout<<"Enter the values for "<<(i+1)<< " row"<<endl;
                for(int j=0;j<7;j++)
                {
                        cin>>b[i][j];
                }
        }
prims(a,b,0,0);
        return 0;
}
```

OUTPUT:


Quincy 2005
Enter the values for 1 row
0
2
0
6
0
4
2
Enter the values for 2 row
2
0
3
8
5
0
7
Enter the values for 3 row
0
3
0
0
7
6
2
Enter the values for 4 row
6
8
0
0
9
4
3
Enter the values for 5 row
0
5
7
9
0
6
9
Enter the values for 6 row
4
0
0
2
5
0
2
Enter the values for 7 row
0
1
2
0
6
3
8

```
The source node: 0
The Destination node: 1
The weight of node: 2
The source node: 0
The Destination node: 6
The weight of node: 2
The source node: 6
The Destination node: 1
The weight of node: 1
The source node: 6
The Destination node: 2
The weight of node: 2
The source node: 1
The Destination node: 2
The weight of node: 3
The source node: 6
The Destination node: 5
The weight of node: 3

Press Enter to return to Quincy...
```

10.Write a program to solve the 0-1 knapsnap problem.

CODE:

```
#include <iostream>
using namespace std;
int knapsack(int v[],int w[],int n, int W)
{
	if(W<0)
	cout<<"\nThe weight of sack is less than zero.";
	if(n<0 || W==0)
		return 0;
	int in=v[n]+knapsack(v,w,n-1,W-w[n]);
	int ex=knapsack(v,w,n-1,W);
	return max(in,ex);
}


int main()
{
	int v[6];
	for(int i=0;i<=5;i++)
	{
	cout<<"\nEnter the value of items "<<i<<" : ";
	cin>>v[i];
```

```
        }
        int w[6];
        for(int i=0;i<=5;i++)
        {
        cout<<"\nEnter the weight of items "<<i<<": ";
        cin>>w[i];
        }
        int W;
        cout<<"\nEnter the maximum weight:";
        cin>>W;
        int n=sizeof(v) / sizeof(v[0]);
        cout<<"Knapsack value is "<<knapsack(v,w,n-1,W);
return 0;
}
```

OUTPUT: