Armstrong number.cpp

```cpp
#include<stdio.h>
#include<math.h>
int main(){
    int num,count=0,sum=0;
    printf("Enter a number: ");
    scanf("%d",&num);
    int onum=num;
    int temp=num;
    while(temp>0){
        temp/=10;
        count++;
    }temp=num;
    while(temp!=0){
        int d=temp%10;
        sum+=pow(d,count);
        temp/=10;
    }
    if(sum==onum){
        printf("%d is a Armstrong Number",onum);
    }else{
        printf("%d is not a Armstrong number",onum);
    }
    return 0;
}
```

Mth max and Nth min.cpp

```c
#include<stdio.h>
int main(){
    int arr[]={12,78,34,42,85};
    int s=sizeof(arr)/sizeof(arr[0]);
    for(int i=0;i<s;i++){
        for(int j=i+1;j<s;j++){
            if(arr[i]>arr[j]){
                int temp=arr[i];
                arr[i]=arr[j];
                arr[j]=temp;
            }
        }
    }
    int m=2,n=3;
    int max=arr[s-m];
    int min=arr[n-1];
    printf("%dth Maximum = %d\n",m,max);
    printf("%dth Minimum = %d\n",n,min);
    int sum=max+min;
    int diff=max-min;
    printf("Sum = %d\n",sum);
    printf("Difference = %d",diff);
    return 0;
}
```

```
C:\Users\babyk\Documents\C
2th Maximum = 78
3th Minimum = 42
Sum = 120
Difference = 36
--------------------------------
Process exited after 0.7865 seconds wit
Press any key to continue . . .
```

Binary Search.cpp

```c
#include<stdio.h>
int main(){
    int arr[]={-2,-4,0,2,4,6,8,10,18,22};
    int n=sizeof(arr)/sizeof(arr[0]);
    int start=0,end=n-1,k=18;
    while(start<end){
        int mid=(start+end)/2;
        if(arr[mid]==k){
            printf("%d found at index %d.",k,mid);
            return 0;
        }else if(arr[mid]<k){
            start=mid+1;
        }else if(arr[mid]>k){
            end=mid-1;
        }
    }
    printf("Element not found");
    return 0;
}
```

Strassen's Matrix Multiplication.cpp

```cpp
#include <stdio.h>
int main() {
    int A[2][2], B[2][2], C[2][2], M1, M2, M3, M4, M5, M6, M7;
    printf("Enter the elements of 2x2 matrix A:\n");
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            printf("A[%d][%d] = ", i, j); scanf("%d", &A[i][j]);
        }
    }printf("Enter the elements of 2x2 matrix B:\n");
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            printf("B[%d][%d] = ", i, j); scanf("%d", &B[i][j]);
        }
    }
    M1 = (A[0][0] + A[1][1]) * (B[0][0] + B[1][1]); // M1 = (a + d)(e + h)
    M2 = (A[1][0] + A[1][1]) * B[0][0];             // M2 = (c + d)e
    M3 = A[0][0] * (B[0][1] - B[1][1]);             // M3 = a(f - h)
    M4 = A[1][1] * (B[1][0] - B[0][0]);             // M4 = d(g - e)
    M5 = (A[0][0] + A[0][1]) * B[1][1];             // M5 = (a + b)h
    M6 = (A[1][0] - A[0][0]) * (B[0][0] + B[0][1]); // M6 = (c - a)(e + f)
    M7 = (A[0][1] - A[1][1]) * (B[1][0] + B[1][1]); // M7 = (b - d)(g + h)
    C[0][0] = M1 + M4 - M5 + M7; // C11 = M1 + M4 - M5 + M7
    C[0][1] = M3 + M5;           // C12 = M3 + M5
    C[1][0] = M2 + M4;           // C21 = M2 + M4
    C[1][1] = M1 - M2 + M3 + M6; // C22 = M1 - M2 + M3 + M6
    printf("Resultant Matrix C (AxB) is:\n");
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            printf("%d ", C[i][j]);
        }printf("\n");
    }return 0;
}
```

pascal triangle.cpp

```
1   #include<stdio.h>
2   int main(){
3       int n,a,k,i,j;
4       printf("enter number of rows:");
5       scanf("%d",&n);
6       for(i=0;i<n;i++){
7           a=1;
8           for(k=1;k<=n-i;k++){
9               printf(" ");
10          }
11          for(j=0;j<=i;j++){
12              printf(" %d ",a);
13              a=a*(i-j)/(j+1);
14          }
15          printf("\n");
16      }
17      return 0;
18  }
```

GCD of 2 numbers.cpp

```c
#include<stdio.h>
#include<math.h>
int main(){
    int n1,n2;
    printf("Enter 2 numbers: ");
    scanf("%d %d",&n1,&n2);
    int a=n1,b=n2;
    while(n2!=0){
        int temp=n1%n2;
        n1=n2;
        n2=temp;
    }
    printf("GCD of %d and %d is %d",a,b,n1);
    return 0;
}
```

Floyd's algorithm.cpp

```cpp
#include <stdio.h>
#define N 4
int main() {
    int graph[N][N] = {
        {0, 5, 10, 15},
        {5, 0, 3, 5},
        {10, 3, 0, 2},
        {15, 5, 2, 0}
    };
    for (int k = 0; k < N; k++) {
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                if (graph[i][k] + graph[k][j] < graph[i][j]) {
                    graph[i][j] = graph[i][k] + graph[k][j];
                }
            }
        }
    }
    printf("Shortest Distances:\n");
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            printf("%d ", graph[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

sumDidits.cpp

```cpp
#include <stdio.h>
int main() {
    int nums[] = {51, 71, 17, 42};
    int n = sizeof(nums) / sizeof(nums[0]);
    int max1[100] = {0}, max2[100] = {0}, maxSum = -1;
    for (int i = 0; i < n; i++) {
        int sum = 0, temp = nums[i];
        while (temp > 0) {
            sum += temp % 10;
            temp /= 10;
        }
        if (nums[i] > max1[sum]) {
            max2[sum] = max1[sum];
            max1[sum] = nums[i];
        } else if (nums[i] > max2[sum]) {
            max2[sum] = nums[i];
        }
    }
    for (int i = 0; i < 100; i++) {
        if (max1[i] != 0 && max2[i] != 0 && max1[i] + max2[i] > maxSum) {
            maxSum = max1[i] + max2[i];
        }
    }
    printf(maxSum != -1 ? "Maximum sum: %d\n" : "No valid pairs found.\n", maxSum);
    return 0;
}
```

liner search.cpp

```c
#include <stdio.h>
int main() {
    int arr[100];
    int n, target, i;
    printf("Enter the number of elements: ");
    scanf("%d", &n);
    printf("Enter %d elements: ", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    printf("Enter the target element: ");
    scanf("%d", &target);
    for (i = 0; i < n; i++) {
        if (arr[i] == target) {
            printf("Element found at index %d\n", i);
            return 0;
        }
    }
    printf("Element not found\n");
    return 0;
}
```

merge sort.cpp

```c
#include<stdio.h>
int main(){
    int a[]={2,8,9,3,4};
    int len1=sizeof(a)/sizeof(a[0]);
    int b[]={-2,5,9,1};
    int len2=sizeof(b)/sizeof(b[0]);
    int m[len1+len2], i,j;
    for(i=0;i<len1;i++){
        m[i]=a[i];
    }
    for(i=0;i<len2;i++){
        m[i+len1]=b[i];
    }
    printf("merge 2 array: ");
    for(i=0;i<len1+len2;i++){
        printf(" %d ",m[i]);
    }
    for(i=0;i<len1+len2;i++){
        for(j=i+1;j<len1+len2;j++){
            if(m[i]>m[j]){
                int temp=m[i];
                m[i]=m[j];
                m[j]=temp;
            }
        }
    }
    printf("\nsorted array: ");
    for(i=0;i<len1+len2;i++){
        printf(" %d ",m[i]);
    }
    return 0;
}
```

Binary Coeffecient.cpp

```cpp
#include <stdio.h>
int main() {
    int n = 2, k = 2;
    int C[n+1][k+1];
    for (int i = 0; i <= n; i++) {
        for (int j = 0; j <= (i < k ? i : k); j++) {
            if (j == 0 || j == i) {
                C[i][j] = 1;
            } else {
                C[i][j] = C[i-1][j-1] + C[i-1][j];
            }
        }
    }
    printf("C(%d, %d) = %d\n", n, k, C[n][k]);
    return 0;
}
```

minimum and maximum value sequency for all the numbers in a list.cpp

```c
#include <stdio.h>
#include <limits.h>
int main() {
    int arr[] = {10, 50, 2, 100, 40, 20};
    int n = sizeof(arr) / sizeof(arr[0]);
    int max = arr[0];
    int min = arr[0];
    for (int i = 0; i < n; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
        if (arr[i] < min) {
            min = arr[i];
        }
    }
    printf("Array: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    printf("Minimum Value: %d\n", min);
    printf("Maximum Value: %d\n", max);
    return 0;
}
```

22.Binomial Coeff.cpp

```c
#include<stdio.h>
int main(){
    int n,r;
    printf("Enter the value of n: ");
    scanf("%d",&n);
    printf("Enter the value of r: ");
    scanf("%d",&r);
    int coff=1;
    for(int i=1;i<=r;i++){
        coff=coff*(n-i+1)/i;
    }
    printf("The binomial coefficient for %d C %d is %d"
    ,n,r,coff);
    return 0;
}
```

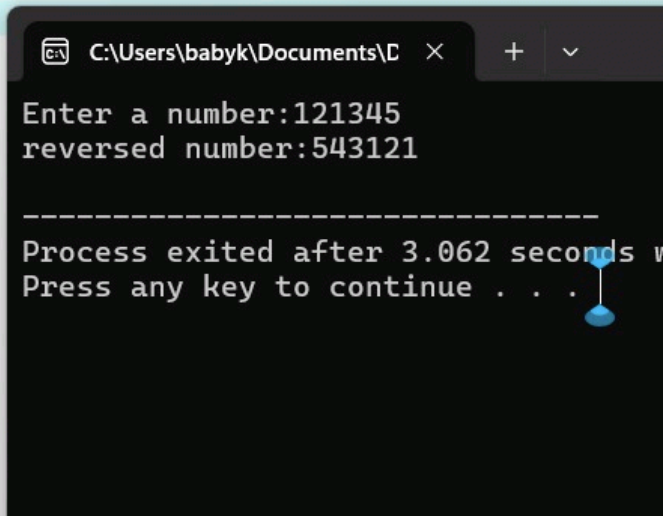
```
Enter the value of n: 4
Enter the value of r: 4
The binomial coefficient for 4 C 4 is 1
--------------------------------
Process exited after 2.566 seconds with retu
Press any key to continue . . .
```

sum of subsets problem using backtracking.cpp

```c
#include <stdio.h>
#define MAX 100
void printSubset(int subset[], int subsetSize) {
    printf("(");
    for (int i = 0; i < subsetSize; i++) {
        printf("%d", subset[i]);
        if (i < subsetSize - 1) {
            printf(",");
        }
    }printf(")\n");
}void findSubsets(int set[], int subset[], int n, int subsetSize, int sum, int targetSum, int index) {
    if (sum == targetSum) {
        printSubset(subset, subsetSize);
        return;
    }if (sum > targetSum || index == n) {
        return;
    }subset[subsetSize] = set[index];
    findSubsets(set, subset, n, subsetSize + 1, sum + set[index], targetSum, index + 1);
    findSubsets(set, subset, n, subsetSize, sum, targetSum, index + 1);
}int main() {
    int set[] = {6, 2, 8, 1, 5}; int n = sizeof(set) / sizeof(set[0]);
    int targetSum = 9; int subset[MAX];
    printf("Subsets with sum %d are:\n", targetSum);
    findSubsets(set, subset, n, 0, 0, targetSum, 0);
    return 0;
```

reverse a number.cpp

```c
#include<stdio.h>
int main(){
    int  num,reverse=0;
    printf("Enter a number:");
    scanf("%d",&num);
    while(num!=0){
        int remainder=num%10;
        reverse=reverse*10+remainder;
        num/=10;
    }
    printf("reversed number:%d\n",reverse);
    return 0;
}
```



Knapsack using Greedy technique.cpp

```cpp
#include <stdio.h>
#define MAX_ITEMS 4
#define MAX_WEIGHT 100
int main() {
    int weights[MAX_ITEMS] = {40, 30, 20, 30};
    int profits[MAX_ITEMS] = {80, 70, 50, 80};
    int knapsackWeight = MAX_WEIGHT;
    int dp[MAX_ITEMS + 1][knapsackWeight + 1];
    for (int i = 0; i <= MAX_ITEMS; i++) {
        for (int w = 0; w <= knapsackWeight; w++) {
            if (i == 0 || w == 0) {
                dp[i][w] = 0;
            } else if (weights[i - 1] <= w) {
                dp[i][w] = (profits[i - 1] + dp[i - 1][w - weights[i - 1]] > dp[i - 1][w])
                           ? profits[i - 1] + dp[i - 1][w - weights[i - 1]]
                           : dp[i - 1][w];
            } else {
                dp[i][w] = dp[i - 1][w];
            }
        }
    }
    printf("Maximum profit in the knapsack: %d\n", dp[MAX_ITEMS][knapsackWeight]);
    return 0;
}
```

Perfect.cpp

```c
#include <stdio.h>
int main() {
    int sum, count = 0, i, j, n;
    printf("Enter the number of perfect numbers to find: ");
    scanf("%d", &n);
    for (i = 1; count < n; i++) {
        sum = 0;
        for (j = 1; j <= i / 2; j++) {
            if (i % j == 0) {
                sum += j;
            }
        }
        if (sum == i) {
            printf("%d ", i);
            count++;
        }
    }
    printf("\n");
    return 0;
}

```

Prim's algorithm.cpp

```c
#include <stdio.h>
#include <limits.h>
#define V 5
int main() {
    int graph[V][V] = {
        {0, 2, 0, 6, 0},
        {2, 0, 3, 8, 5},
        {0, 3, 0, 0, 7},
        {6, 8, 0, 0, 9},
        {0, 5, 7, 9, 0}
    }; int parent[V], key[V], mstSet[V];
    for (int i = 0; i < V; i++) {
        key[i] = INT_MAX;
        mstSet[i] = 0;
    }
    key[0] = 0; parent[0] = -1;
    for (int count = 0; count < V - 1; count++) {
        int min = INT_MAX, u;
        for (int v = 0; v < V; v++)
            if (!mstSet[v] && key[v] < min) {
                min = key[v]; u = v;
            }
        mstSet[u] = 1;
        for (int v = 0; v < V; v++) {
            if (graph[u][v] && !mstSet[v] && graph[u][v] < key[v]) {
                parent[v] = u; key[v] = graph[u][v];
            }
        }
    } printf("Edge \tWeight\n");
    for (int i = 1; i < V; i++)
        printf("%d - %d \t%d \n", parent[i], i, graph[parent[i]][i]);
    return 0;
}
```

Selection sort.cpp

```cpp
#include<stdio.h>
int main(){
    int arr[]={64,25,12,22,11};
    int n=sizeof(arr)/sizeof(arr[0]);
    printf("Intial Array:\n");
    for(int i=0;i<n;i++){
        printf("%d ",arr[i]);
    }
    int i,j,minIndex,temp;
    for(i=0;i<=n-1;i++){
        minIndex=i;
        for(j=i+1;j<=n-1;j++){
            if(arr[j]<arr[minIndex]){
                minIndex=j;
            }
        }
        if(minIndex!=i){
            temp=arr[i];
            arr[i]=arr[minIndex];
            arr[minIndex]=temp;
        }
    }
    printf("\nSelection sort: \n");
    for(i=0;i<n;i++){
        printf("%d ",arr[i]);
    }
    return 0;
}
```

perfect number.cpp

```c
#include<stdio.h>
int main(){
    int n,i,j,sum=0,num;
    printf("enter the number:");
    scanf("%d",&n);
    num=n;
    for(i=1;i<n;i++)
    {
        if(n%i==0){
            sum=sum+i;
        }
    }
    if(sum==num){
        printf("perfect number:");
    }else{
        printf("not a perfect number:");
    }
    return 0;
}
```

even sum fibonacci.cpp

```c
#include <stdio.h>
int main() {
    int n, i;
    int fib[100];
    int evenSum = 0;
    printf("Enter the value of n: ");
    scanf("%d", &n);
    fib[0] = 0;
    fib[1] = 1;
    for (i = 2; i <= 8; i++) {
        fib[i] = fib[i - 1] + fib[i - 2];
    }
    for (i = 0; i <= 8; i += 2) {
        evenSum += fib[i];
    }
    printf("Sum of Fibonacci numbers at even indices: %d\n", evenSum);
    return 0;
}
```

String Palindrome.cpp

```c
#include<stdio.h>
#include<string.h>
int main(){
    char str[100];
    char rev[100];
    printf("Enter a String: ");
    scanf("%s",str);
    int len=strlen(str);
    rev[len]='\0';
    for(int i=len-1;i>=0;i--){
        rev[len-i-1]=str[i];
    }
    if(strcmp(str,rev)==0){
        printf("%s is a Palindrome String",str);
    }else{
        printf("%s is not a Palindrome string",str);
    }
    return 0;
}
```

String Palindrome.cpp

```cpp
#include<stdio.h>
#include<string.h>
int main(){
    char str[100];
    char rev[100];
    printf("Enter a String: ");
    scanf("%s",str);
    int len=strlen(str);
    rev[len]='\0';
    for(int i=len-1;i>=0;i--){
        rev[len-i-1]=str[i];
    }
    if(strcmp(str,rev)==0){
        printf("%s is a Palindrome String",str);
    }else{
        printf("%s is not a Palindrome string",str);
    }
    return 0;
}
```

inset a number in a list.cpp

```cpp
#include<stdio.h>
int main(){
    int a[6]={1,2,3,4,5};
    int p=3,i,x;
    printf("enter the element to insert:");
    scanf("%d",&x);
    for(i=5;i>=p;i--){
        a[i]=a[i-1];
    }
    a[p-1]=x;
    for(i=0;i<6;i++){
        printf("element %d: %d\n",i+1,a[i]);
    }
    return 0;
}
```

Prim's algorithm.cpp

```cpp
#include <stdio.h>
#include <limits.h>
#define V 5
int main() {
    int graph[V][V] = {
        {0, 2, 0, 6, 0},
        {2, 0, 3, 8, 5},
        {0, 3, 0, 0, 7},
        {6, 8, 0, 0, 9},
        {0, 5, 7, 9, 0}
    }; int parent[V], key[V], mstSet[V];
    for (int i = 0; i < V; i++) {
        key[i] = INT_MAX;
        mstSet[i] = 0;
    } key[0] = 0; parent[0] = -1;
    for (int count = 0; count < V - 1; count++) {
        int min = INT_MAX, u;
        for (int v = 0; v < V; v++)
            if (!mstSet[v] && key[v] < min) {
                min = key[v]; u = v;
            } mstSet[u] = 1;
        for (int v = 0; v < V; v++) {
            if (graph[u][v] && !mstSet[v] && graph[u][v] < key[v]) {
                parent[v] = u; key[v] = graph[u][v];
            }
        }
    } printf("Edge \tWeight\n");
    for (int i = 1; i < V; i++)
        printf("%d - %d \t%d \n", parent[i], i, graph[parent[i]][i]);
    return 0;
}
```

Reversed string.cpp

```c
#include<stdio.h>
#include<string.h>
int main(){
    char str[100];
    printf("Enter a string: ");
    scanf("%s",str);
    int length=strlen(str);
    printf("Reversed String: ");
    for(int i=length-1;i>=0;i--){
        printf("%c",str[i]);
    }
    return 0;
}
```

2 minimum values.cpp

```cpp
#include <stdio.h>
#include <limits.h>
int main() {
    int arr[] = {3, 5, -4, 1, 8, 2, 0, 4};
    int n = sizeof(arr) / sizeof(arr[0]);
    int min1 = arr[0];
    int min2 = arr[0];
    for (int i = 0; i < n; i++) {
        if (arr[i] < min1) {
            min2 = min1;
            min1 = arr[i];
        } else if (arr[i] < min2 && arr[i] != min1) {
            min2 = arr[i];
        }
    }
    if (min1 != INT_MAX && min2 != INT_MAX) {
        printf("Output (%d, %d)\n", min1, min2);
    } else {
        printf("Not enough unique values.\n");
    }
    return 0;
}
```

BST.cpp

```c
#include <stdio.h>
#include <limits.h>
int main() {
    int n, keys[n], freq[n];
    printf("Enter the number of keys: ");
    scanf("%d", &n);
    printf("Enter keys: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &keys[i]);
    }
    printf("Enter frequencies: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &freq[i]);
    }
    int cost[n][n];
    for (int i = 0; i < n; i++) {
        cost[i][i] = freq[i];
    }
    for (int length = 2; length <= n; length++) {
        for (int i = 0; i <= n - length; i++) {
            int j = i + length - 1;
            cost[i][j] = INT_MAX;
            for (int r = i; r <= j; r++) {
                int leftCost = (r > i) ? cost[i][r - 1] : 0;
                int rightCost = (r < j) ? cost[r + 1][j] : 0;
                int totalCost = leftCost + rightCost + freq[i] + freq[j];
                if (totalCost < cost[i][j]) {
                    cost[i][j] = totalCost;
                }
            }
        }
    }printf("Minimum cost of the BST is: %d\n", cost[0][n - 1]);
    return 0;
```

pattern pyramid.cpp

```c
#include <stdio.h>
int main() {
    int n, i, j, k;
    printf("Enter the number of rows: ");
    scanf("%d", &n);
    for (i = 1; i <= n; i++) {
        for (j = i; j < n; j++) {
            printf("    ");
        }
        for (k = 1; k <= i; k++) {
            printf("%d         ", k);
        }
        printf("\n");
    }
    return 0;
}
```

Permutations.cpp

```cpp
#include <stdio.h>
int main() {
    int arr[] = {1, 2, 3};
    int n = sizeof(arr) / sizeof(arr[0]);
    int i, j, k;
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            for (k = 0; k < n; k++) {
                if (i != j && j != k && i != k) {
                    printf("[%d, %d, %d]\n", arr[i], arr[j], arr[k]);
                }
            }
        }
    }
    return 0;
}
```

String Palindrome.cpp

```c
#include<stdio.h>
#include<string.h>
int main(){
    char str[100];
    char rev[100];
    printf("Enter a String: ");
    scanf("%s",str);
    int len=strlen(str);
    rev[len]='\0';
    for(int i=len-1;i>=0;i--){
        rev[len-i-1]=str[i];
    }
    if(strcmp(str,rev)==0){
        printf("%s is a Palindrome String",str);
    }else{
        printf("%s is not a Palindrome string",str);
    }
    return 0;
}
```

Subsets.cpp

```c
#include <stdio.h>
#define MAX_SIZE 100
int main() {
    int nums[] = {1, 2, 3};
    int n = sizeof(nums) / sizeof(nums[0]);
    int totalSubsets = 1 << n;
    for (int i = 0; i < totalSubsets; i++) {
        printf("[");
        for (int j = 0; j < n; j++) {
            if (i & (1 << j)) {
                printf("%d", nums[j]);
                if (j < n - 1) printf(", ");
            }
        }
        printf("],\n");
    }
    return 0;
}
```

Hamiltonian circuit Using backtracking method.cpp

```cpp
#include <stdio.h>
#define N 5
int graph[N][N] = {
    {0, 1, 1, 1, 0},
    {1, 0, 1, 1, 1},
    {1, 1, 0, 0, 1},
    {1, 1, 0, 0, 1},
    {0, 1, 1, 1, 0}
}; int path[N], visited[N];
int main() {
    int i, pos = 1; path[0] = 0; visited[0] = 1;
    while (pos < N) {
        for (i = 1; i < N; i++) {
            if (graph[path[pos - 1]][i] && !visited[i]) {
                path[pos] = i;
                visited[i] = 1;
                pos++; break;
            }
        }
        if (i == N) {
            visited[path[--pos]] = 0;
        }
    }
    if (graph[path[pos - 1]][path[0]]) {
        printf("Hamiltonian Circuit: ");
        for (i = 0; i < N; i++) printf("%d ", path[i]);
        printf("%d\n", path[0]);
    } else {
        printf("No Hamiltonian Circuit exists\n");
    } return 0;
}
```

Bubble sort.cpp

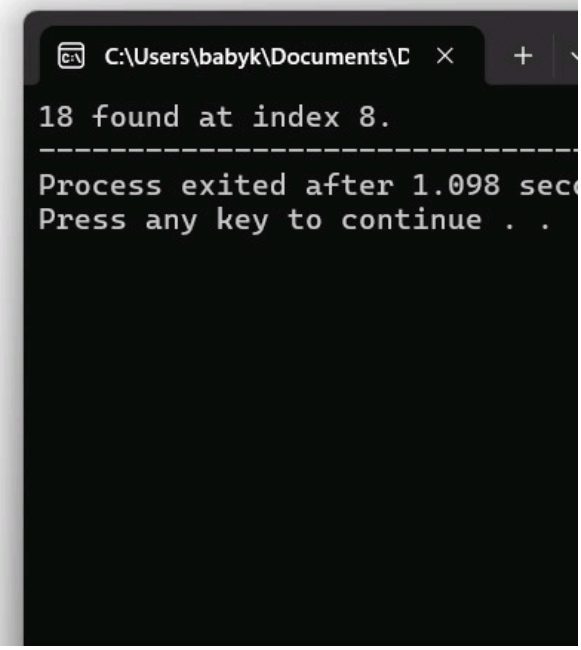
```c
#include<stdio.h>
int main(){
    int arr[]={64,34,25,12,22};
    int n=sizeof(arr)/sizeof(arr[0]);
    int i,j,temp;
    printf("Intial Array\n");
    for(i=0;i<n;i++){
        printf("%d ",arr[i]);
    }
    for(i=0;i<n;i++){
        for(j=i+1;j<n;j++){
            if(arr[i]>arr[j]){
                temp=arr[i];
                arr[i]=arr[j];
                arr[j]=temp;
            }
        }
    }
    printf("\nAfter sorting\n");
    for(i=0;i<n;i++){
        printf("%d ",arr[i]);
    }
    return 0;
}
```

Knapsack using Greedy technique.cpp

```cpp
#include <stdio.h>
#define MAX_ITEMS 4
#define MAX_WEIGHT 100
int main() {
    int weights[MAX_ITEMS] = {40, 30, 20, 30};
    int profits[MAX_ITEMS] = {80, 70, 50, 80};
    int knapsackWeight = MAX_WEIGHT;
    int dp[MAX_ITEMS + 1][knapsackWeight + 1];
    for (int i = 0; i <= MAX_ITEMS; i++) {
        for (int w = 0; w <= knapsackWeight; w++) {
            if (i == 0 || w == 0) {
                dp[i][w] = 0;
            } else if (weights[i - 1] <= w) {
                dp[i][w] = (profits[i - 1] + dp[i - 1][w - weights[i - 1]] > dp[i - 1][w])
                           ? profits[i - 1] + dp[i - 1][w - weights[i - 1]]
                           : dp[i - 1][w];
            } else {
                dp[i][w] = dp[i - 1][w];
            }
        }
    }
    printf("Maximum profit in the knapsack: %d\n", dp[MAX_ITEMS][knapsackWeight]);
    return 0;
}
```

Binary Search.cpp

```c
#include<stdio.h>
int main(){
    int arr[]={-2,-4,0,2,4,6,8,10,18,22};
    int n=sizeof(arr)/sizeof(arr[0]);
    int start=0,end=n-1,k=18;
    while(start<end){
        int mid=(start+end)/2;
        if(arr[mid]==k){
            printf("%d found at index %d.",k,mid);
            return 0;
        }else if(arr[mid]<k){
            start=mid+1;
        }else if(arr[mid]>k){
            end=mid-1;
        }
    }
    printf("Element not found");
    return 0;
}
```



even sum fibonacci.cpp

```c
#include <stdio.h>
int main() {
    int n, i;
    int fib[100];
    int evenSum = 0;
    printf("Enter the value of n: ");
    scanf("%d", &n);
    fib[0] = 0;
    fib[1] = 1;
    for (i = 2; i <= 8; i++) {
        fib[i] = fib[i - 1] + fib[i - 2];
    }
    for (i = 0; i <= 8; i += 2) {
        evenSum += fib[i];
    }
    printf("Sum of Fibonacci numbers at even indices: %d\n", evenSum);
    return 0;
}
```

factorial.cpp

```c
#include <stdio.h>
int factorial(int n) {
    if (n == 0 || n == 1) {
        return 1;
    }
    else {
        return n * factorial(n - 1);
    }
}
int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    if (num < 0) {
        printf("Factorial of a negative number is not defined.\n");
    } else {
        printf("Factorial of %d is %d\n", num, factorial(num));
    }
    return 0;
}
```