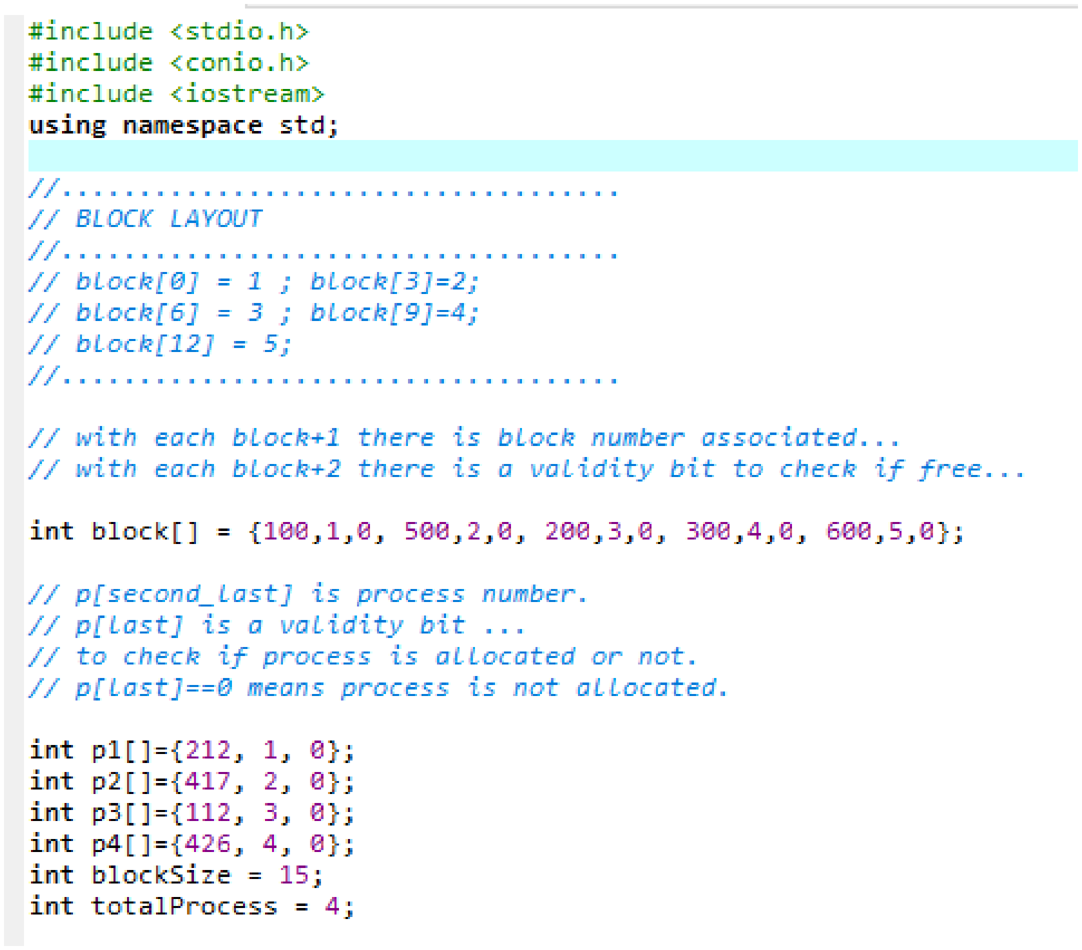
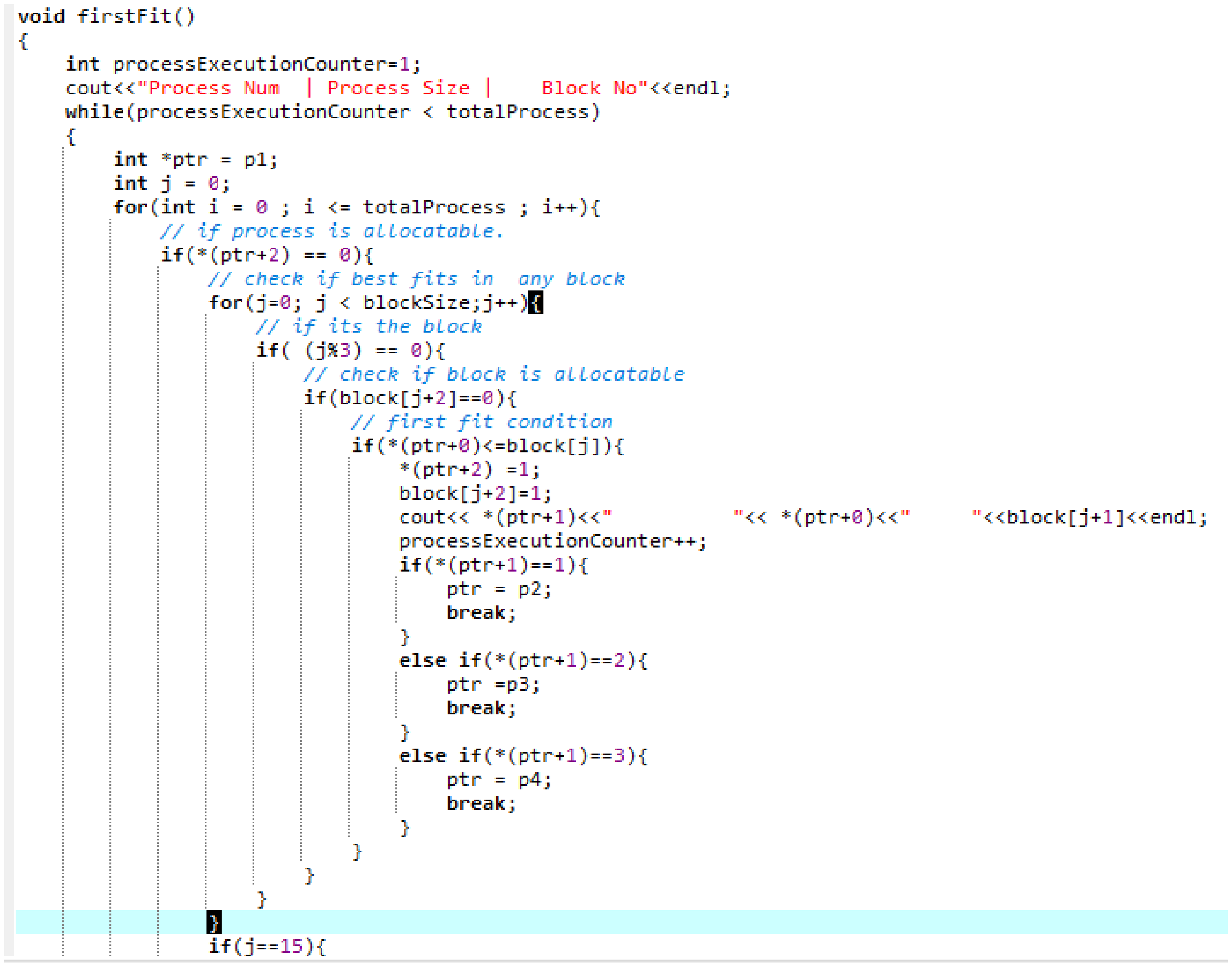
Code Screenshots:

***global variables (used in all three algorithms)**

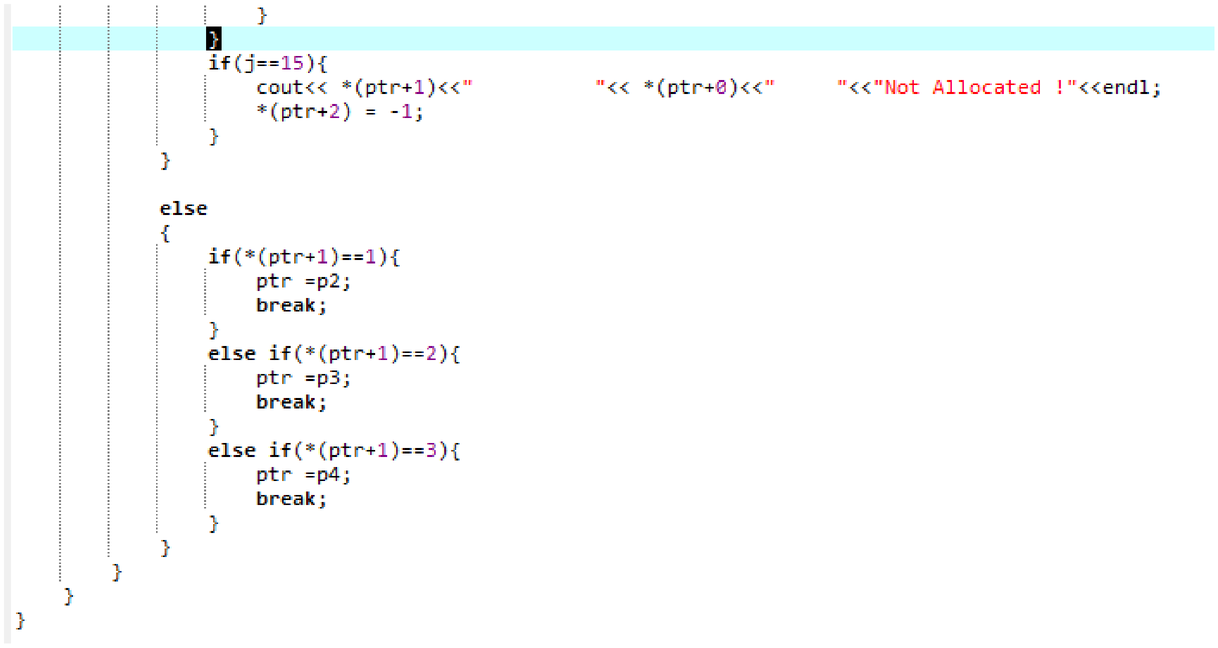


```
#include <stdio.h>
#include <conio.h>
#include <iostream>
using namespace std;

//..................................
// BLOCK LAYOUT
//..................................
// block[0] = 1 ; block[3]=2;
// block[6] = 3 ; block[9]=4;
// block[12] = 5;
//..................................

// with each block+1 there is block number associated...
// with each block+2 there is a validity bit to check if free...

int block[] = {100,1,0, 500,2,0, 200,3,0, 300,4,0, 600,5,0};

// p[second_last] is process number.
// p[last] is a validity bit ...
// to check if process is allocated or not.
// p[last]==0 means process is not allocated.

int p1[]={212, 1, 0};
int p2[]={417, 2, 0};
int p3[]={112, 3, 0};
int p4[]={426, 4, 0};
int blockSize = 15;
int totalProcess = 4;
```

First Fit:



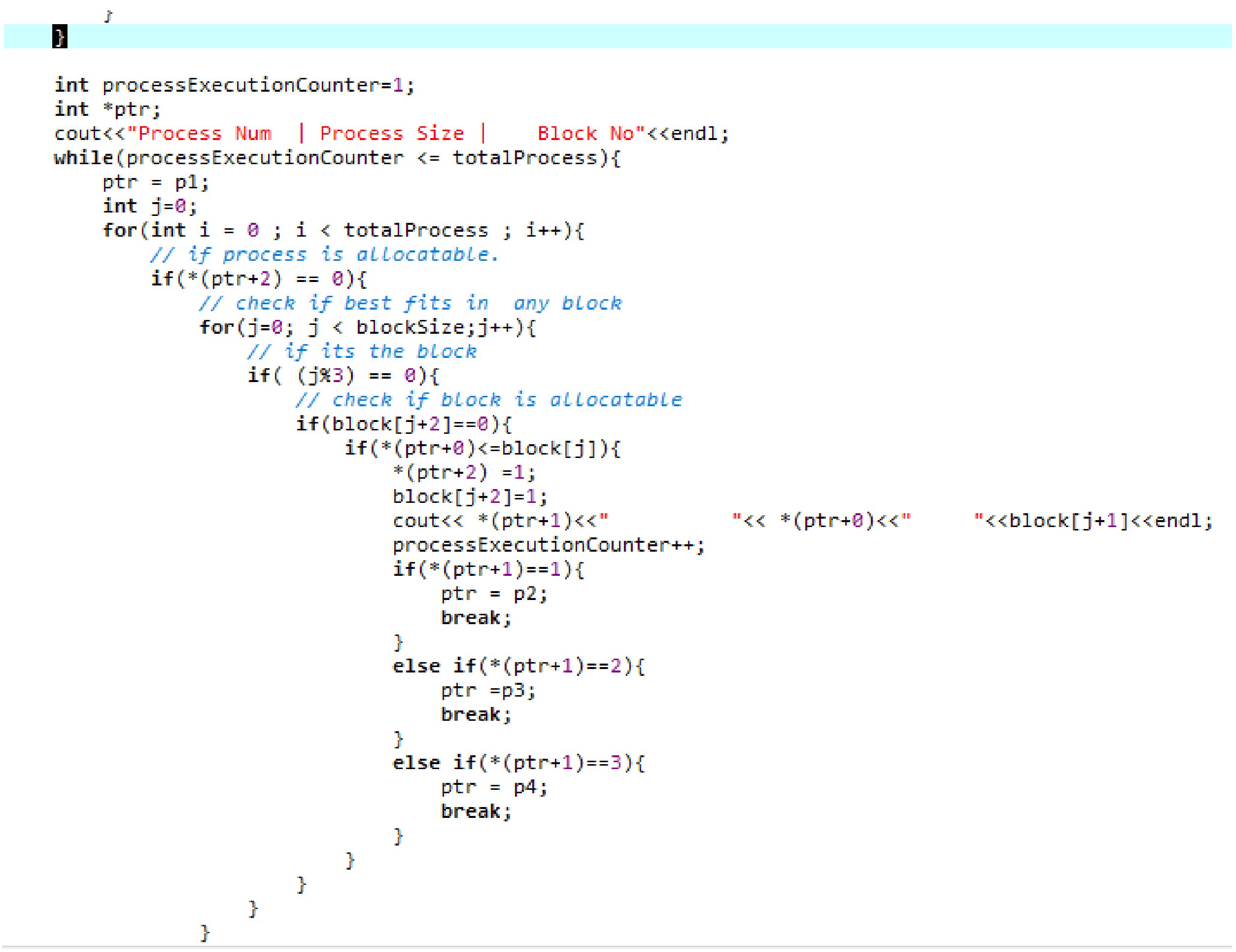
```
void firstFit()
{
    int processExecutionCounter=1;
    cout<<"Process Num  | Process Size |    Block No"<<endl;
    while(processExecutionCounter < totalProcess)
    {
        int *ptr = p1;
        int j = 0;
        for(int i = 0 ; i <= totalProcess ; i++){
            // if process is allocatable.
            if(*(ptr+2) == 0){
                // check if best fits in  any block
                for(j=0; j < blockSize;j++){
                    // if its the block
                    if( (j%3) == 0){
                        // check if block is allocatable
                        if(block[j+2]==0){
                            // first fit condition
                            if(*(ptr+0)<=block[j]){
                                *(ptr+2) =1;
                                block[j+2]=1;
                                cout<< *(ptr+1)<<"           "<< *(ptr+0)<<"     "<<block[j+1]<<endl;
                                processExecutionCounter++;
                                if(*(ptr+1)==1){
                                    ptr = p2;
                                    break;
                                }
                                else if(*(ptr+1)==2){
                                    ptr =p3;
                                    break;
                                }
                                else if(*(ptr+1)==3){
                                    ptr = p4;
                                    break;
                                }
                            }
                        }
                    }
                }
                if(j==15){
```



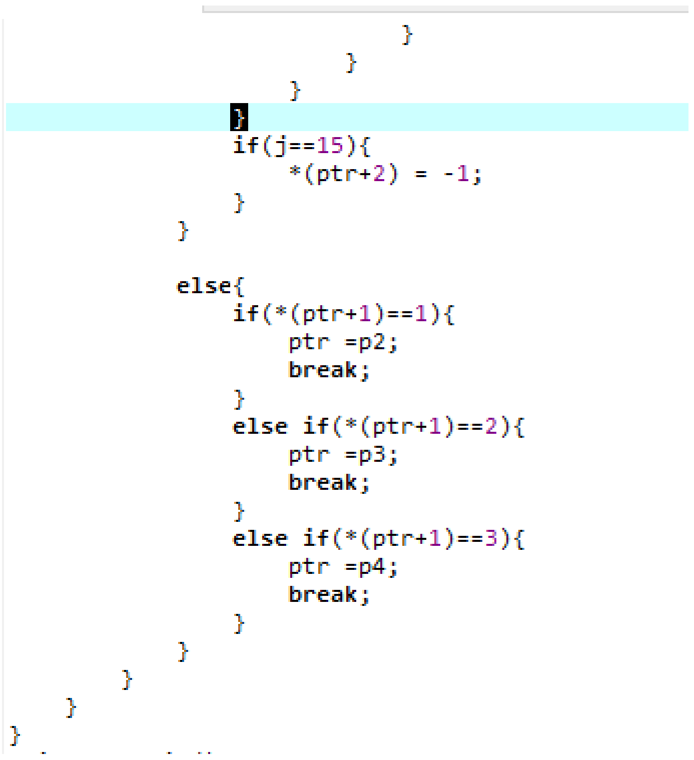
```
                    }
                }
                if(j==15){
                    cout<< *(ptr+1)<<"          "<< *(ptr+0)<<"     "<<"Not Allocated !"<<endl;
                    *(ptr+2) = -1;
                }
            }

            else
            {
                if(*(ptr+1)==1){
                    ptr =p2;
                    break;
                }
                else if(*(ptr+1)==2){
                    ptr =p3;
                    break;
                }
                else if(*(ptr+1)==3){
                    ptr =p4;
                    break;
                }
            }
        }
    }
}
```

Best Fit:

```
void bestFit()
{
    // arranging blocks in ascending order..
    for(int i = 0 ;i<blockSize-1;i++){
        if((i%3)==0){
            for(int j = 0 ;j<blockSize-i-3;j++){
                if ((j%3)==0){
                    if(block[j]>block[j+3]){
                        int blockSize= block[j];
                        int blockNum = block[j+1];
                        int validityBit = block[j+2];

                        block[j]=block[j+3];
                        block[j+1]=block[j+4];
                        block[j+2]=block[j+5];

                        block[j+3]=blockSize;
                        block[j+4]=blockNum;
                        block[j+5]=validityBit;
                    }
                }
            }
        }
    }
```

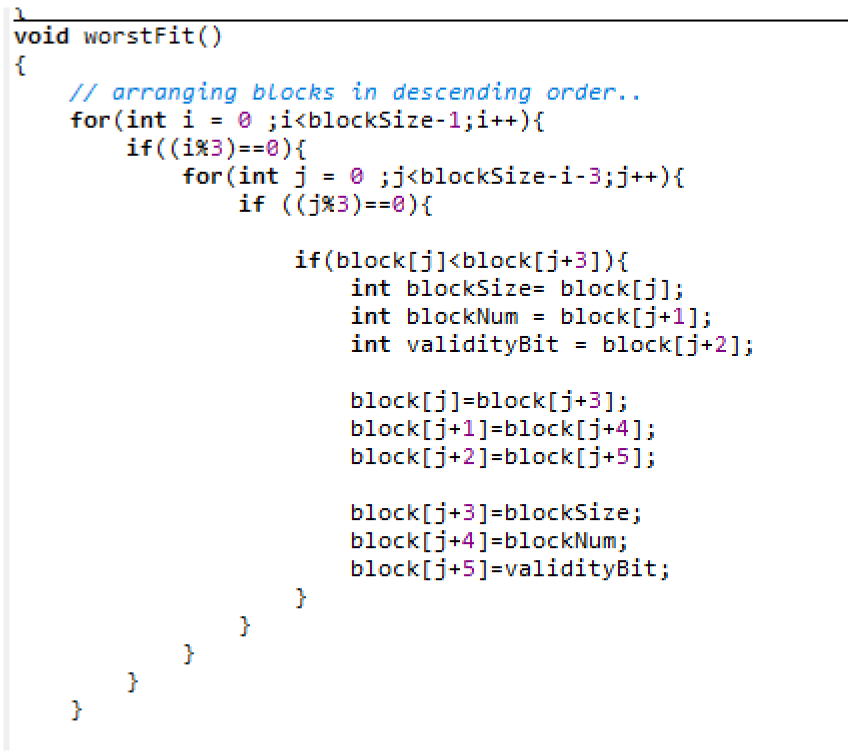


```
        }
    }

    int processExecutionCounter=1;
    int *ptr;
    cout<<"Process Num  | Process Size |    Block No"<<endl;
    while(processExecutionCounter <= totalProcess){
        ptr = p1;
        int j=0;
        for(int i = 0 ; i < totalProcess ; i++){
            // if process is allocatable.
            if(*(ptr+2) == 0){
                // check if best fits in  any block
                for(j=0; j < blockSize;j++){
                    // if its the block
                    if( (j%3) == 0){
                        // check if block is allocatable
                        if(block[j+2]==0){
                            if(*(ptr+0)<=block[j]){
                                *(ptr+2) =1;
                                block[j+2]=1;
                                cout<< *(ptr+1)<<"          "<< *(ptr+0)<<"     "<<block[j+1]<<endl;
                                processExecutionCounter++;
                                if(*(ptr+1)==1){
                                    ptr = p2;
                                    break;
                                }
                                else if(*(ptr+1)==2){
                                    ptr =p3;
                                    break;
                                }
                                else if(*(ptr+1)==3){
                                    ptr = p4;
                                    break;
                                }
                            }
                        }
                    }
                }
```



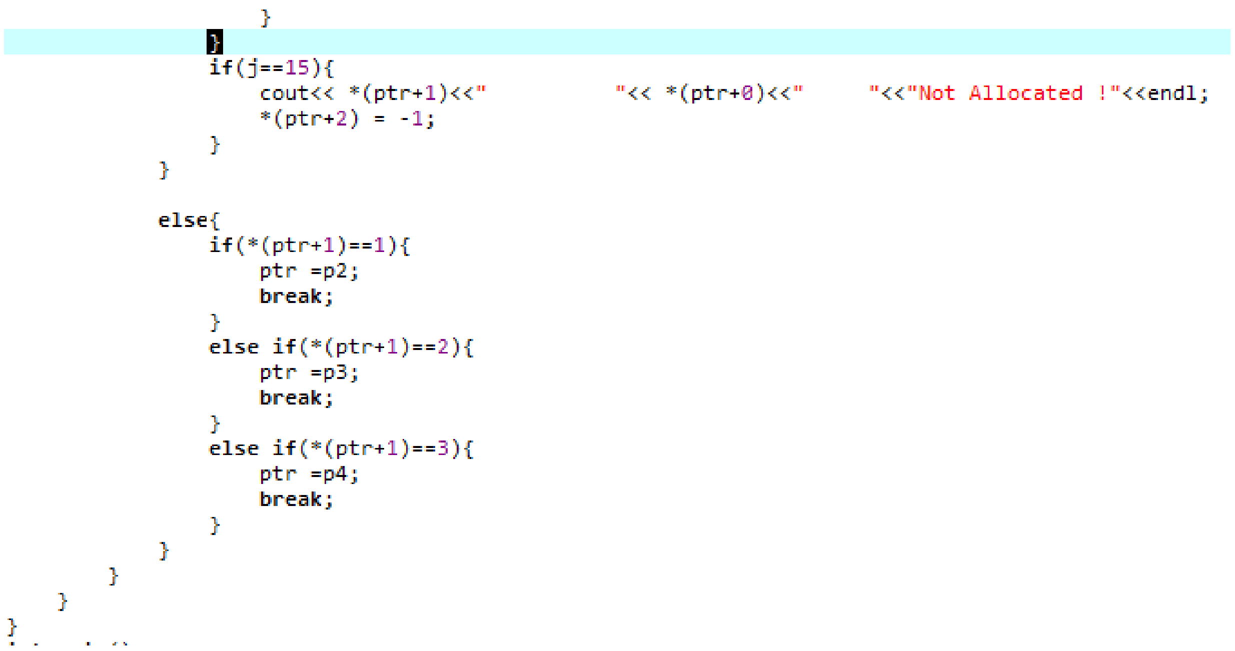
```
                            }
                        }
                    }
                }
                if(j==15){
                    *(ptr+2) = -1;
                }
            }

            else{
                if(*(ptr+1)==1){
                    ptr =p2;
                    break;
                }
                else if(*(ptr+1)==2){
                    ptr =p3;
                    break;
                }
                else if(*(ptr+1)==3){
                    ptr =p4;
                    break;
                }
            }
        }
    }
}
```

Worst Fit:



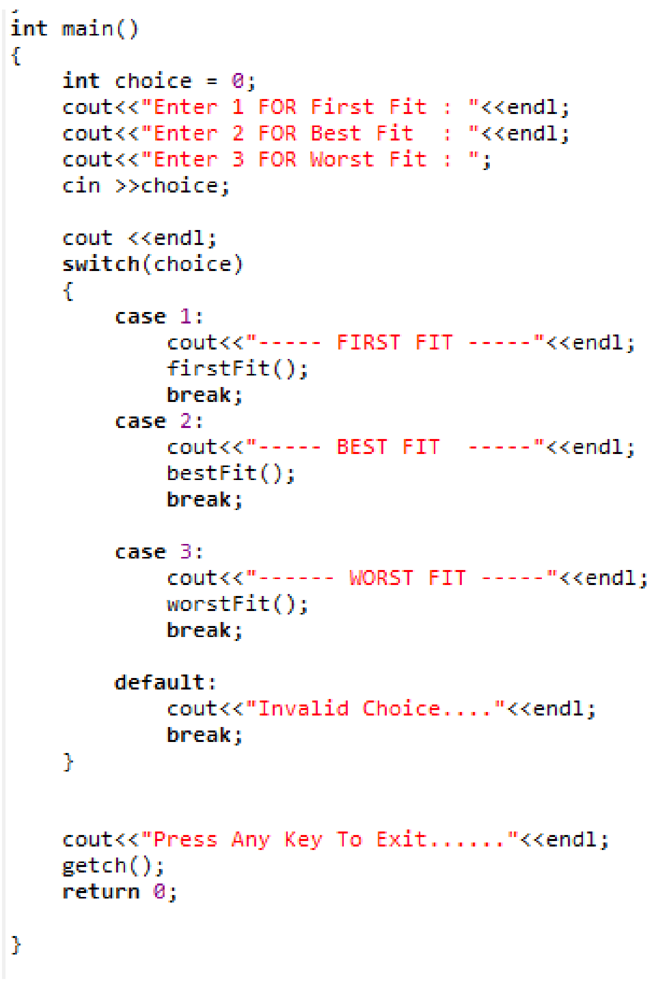
```
}
void worstFit()
{
    // arranging blocks in descending order..
    for(int i = 0 ;i<blockSize-1;i++){
        if((i%3)==0){
            for(int j = 0 ;j<blockSize-i-3;j++){
                if ((j%3)==0){

                    if(block[j]<block[j+3]){
                        int blockSize= block[j];
                        int blockNum = block[j+1];
                        int validityBit = block[j+2];

                        block[j]=block[j+3];
                        block[j+1]=block[j+4];
                        block[j+2]=block[j+5];

                        block[j+3]=blockSize;
                        block[j+4]=blockNum;
                        block[j+5]=validityBit;
                    }
                }
            }
        }
    }
```

```cpp
int processExecutionCounter=1;
int *ptr;
cout<<"Process Num  | Process Size |    Block No"<<endl;
while(processExecutionCounter < totalProcess){
    ptr = p1;
    int j=0;
    for(int i = 0 ; i < totalProcess ; i++){
        // if process is allocatable.
        if(*(ptr+2) == 0){
            // check if best fits in  any block
            for(j=0; j < blockSize;j++){
                // if its the block
                if( (j%3) == 0){
                    // check if block is allocatable
                    if(block[j+2]==0){
                        if(*(ptr+0)<=block[j]){
                            *(ptr+2) =1;
                            block[j+2]=1;
                            cout<< *(ptr+1)<<"          "<< *(ptr+0)<<"     "<<block[j+1]<<endl;

                            processExecutionCounter++;

                            if(*(ptr+1)==1){
                                ptr = p2;
                                break;
                            }
                            else if(*(ptr+1)==2){
                                ptr =p3;
                                break;
                            }
                            else if(*(ptr+1)==3){
                                ptr = p4;
                                break;
                            }
                        }
                    }
                }
            }
            if(j==15){
```



```
                    }
                }
                if(j==15){
                    cout<< *(ptr+1)<<"          "<< *(ptr+0)<<"     "<<"Not Allocated !"<<endl;
                    *(ptr+2) = -1;
                }
            }

            else{
                if(*(ptr+1)==1){
                    ptr =p2;
                    break;
                }
                else if(*(ptr+1)==2){
                    ptr =p3;
                    break;
                }
                else if(*(ptr+1)==3){
                    ptr =p4;
                    break;
                }
            }
        }
    }
}
```

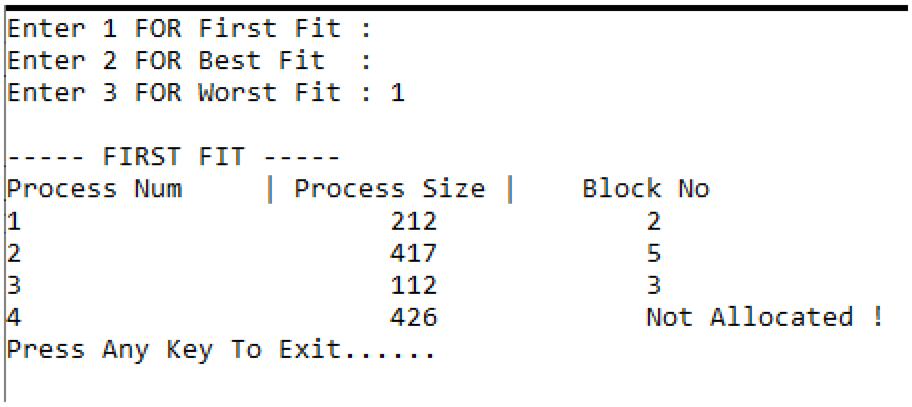
Main:



```
int main()
{
    int choice = 0;
    cout<<"Enter 1 FOR First Fit : "<<endl;
    cout<<"Enter 2 FOR Best Fit  : "<<endl;
    cout<<"Enter 3 FOR Worst Fit : ";
    cin >>choice;

    cout <<endl;
    switch(choice)
    {
        case 1:
            cout<<"----- FIRST FIT -----"<<endl;
            firstFit();
            break;
        case 2:
            cout<<"----- BEST FIT  -----"<<endl;
            bestFit();
            break;

        case 3:
            cout<<"------ WORST FIT -----"<<endl;
            worstFit();
            break;

        default:
            cout<<"Invalid Choice...."<<endl;
            break;
    }


    cout<<"Press Any Key To Exit......"<<endl;
    getch();
    return 0;

}
```

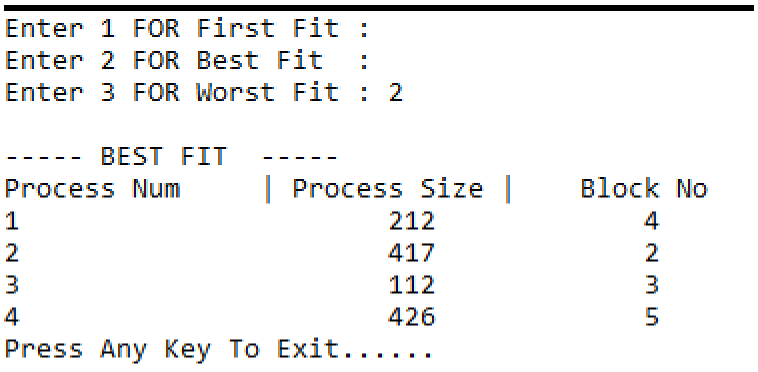
Code Screenshots:

First fit



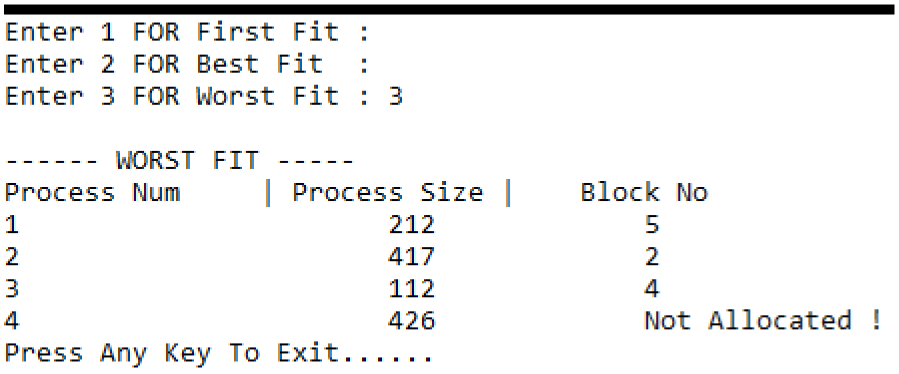
```
Enter 1 FOR First Fit :
Enter 2 FOR Best Fit  :
Enter 3 FOR Worst Fit : 1

----- FIRST FIT -----
Process Num      | Process Size |    Block No
1                        212            2
2                        417            5
3                        112            3
4                        426            Not Allocated !
Press Any Key To Exit......
```

Best fit



```
Enter 1 FOR First Fit :
Enter 2 FOR Best Fit  :
Enter 3 FOR Worst Fit : 2

----- BEST FIT  -----
Process Num      | Process Size |    Block No
1                        212            4
2                        417            2
3                        112            3
4                        426            5
Press Any Key To Exit......
```

Worst fit



```
Enter 1 FOR First Fit :
Enter 2 FOR Best Fit  :
Enter 3 FOR Worst Fit : 3

------ WORST FIT -----
Process Num      | Process Size |    Block No
1                        212            5
2                        417            2
3                        112            4
4                        426            Not Allocated !
Press Any Key To Exit......
```