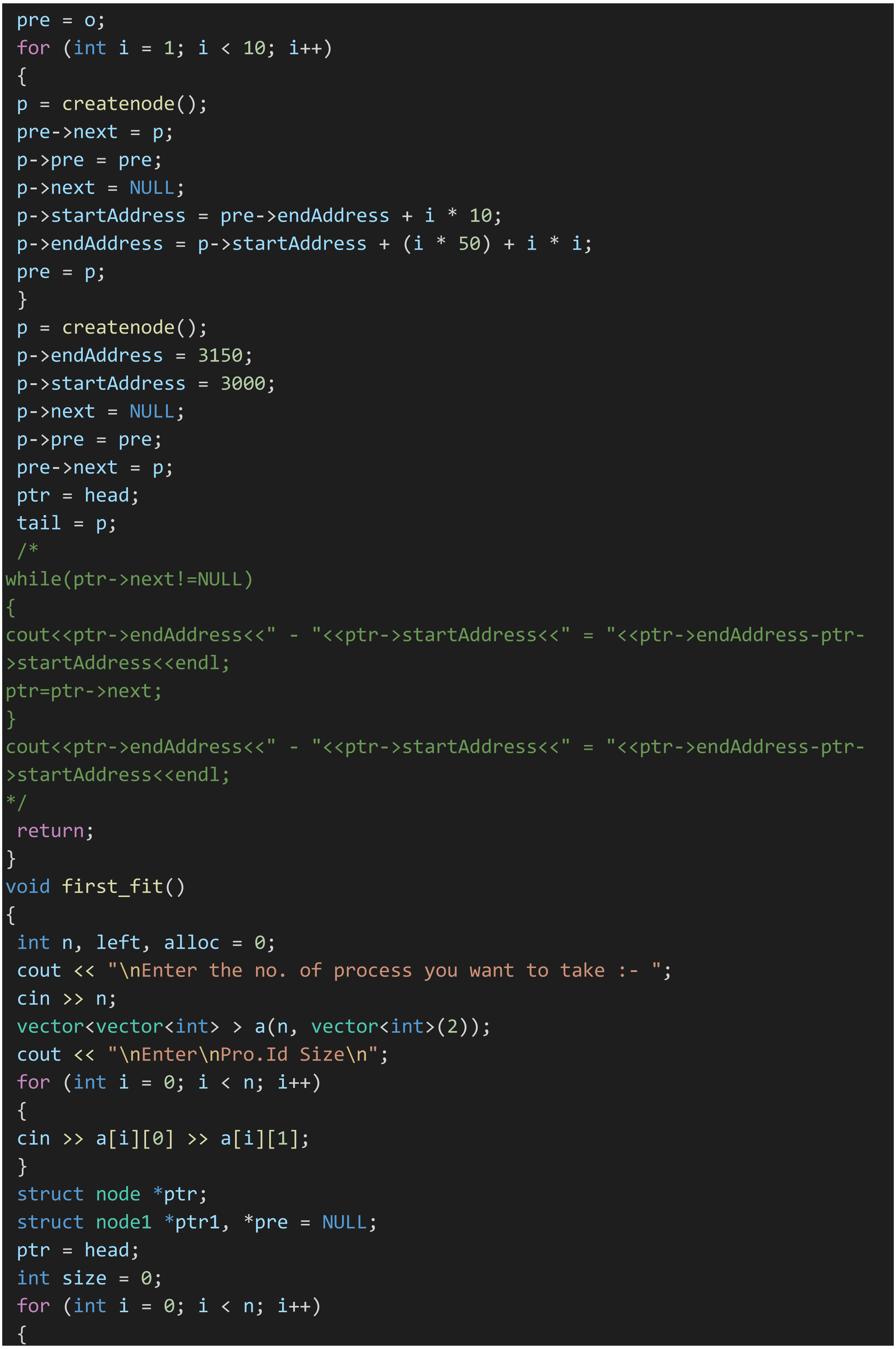
OS ASSIGNMENT 5

NAME:SOURABH PATEL

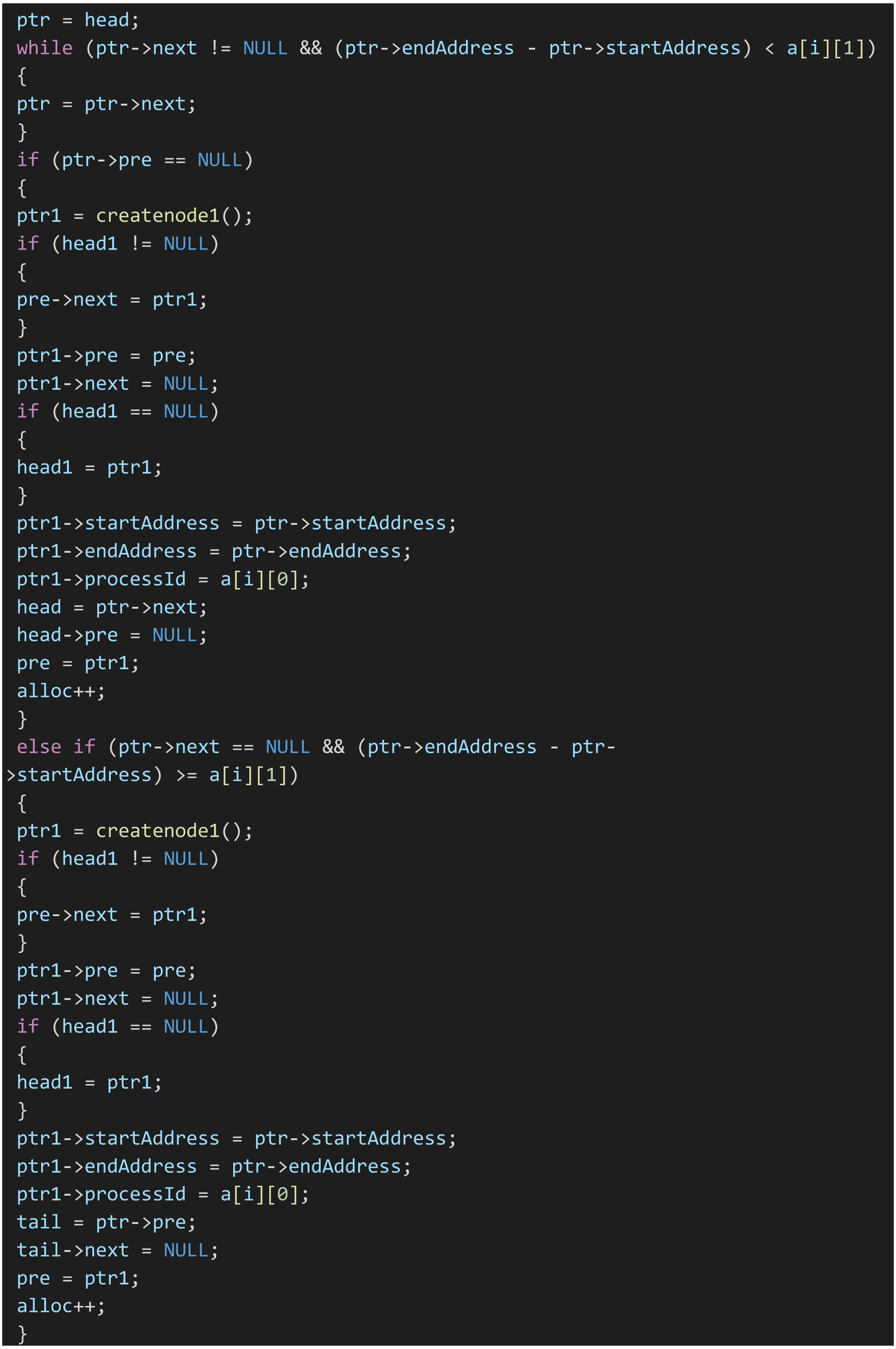
ADMISSION NO:U19CS082

1. **To implement first fit, best fit and worst fit storage allocation algorithms for memory management.**

```
#include <bits/stdc++.h>
using namespace std;
struct node //vacant linked list
{
 int startAddress;
 int endAddress;
 struct node *next;
 struct node *pre;
};
struct node1 //allocated linked list
{
 int startAddress;
 int endAddress;
 int processId;
 struct node1 *next;
 struct node1 *pre;
};
struct node *head, *tail;
struct node1 *head1 = NULL, *tail1;
struct node *createnode()
{
 struct node *t;
 t = (struct node *)malloc(sizeof(struct node));
 return (t);
}
struct node1 *createnode1()
{
 struct node1 *t;
 t = (struct node1 *)malloc(sizeof(struct node1));
 return (t);
}
void memory()
{
 struct node *o, *p, *pre = NULL, *ptr;
 o = createnode();
 o->startAddress = 10;
 o->endAddress = 90;
 o->pre = NULL;
 head = o;
```

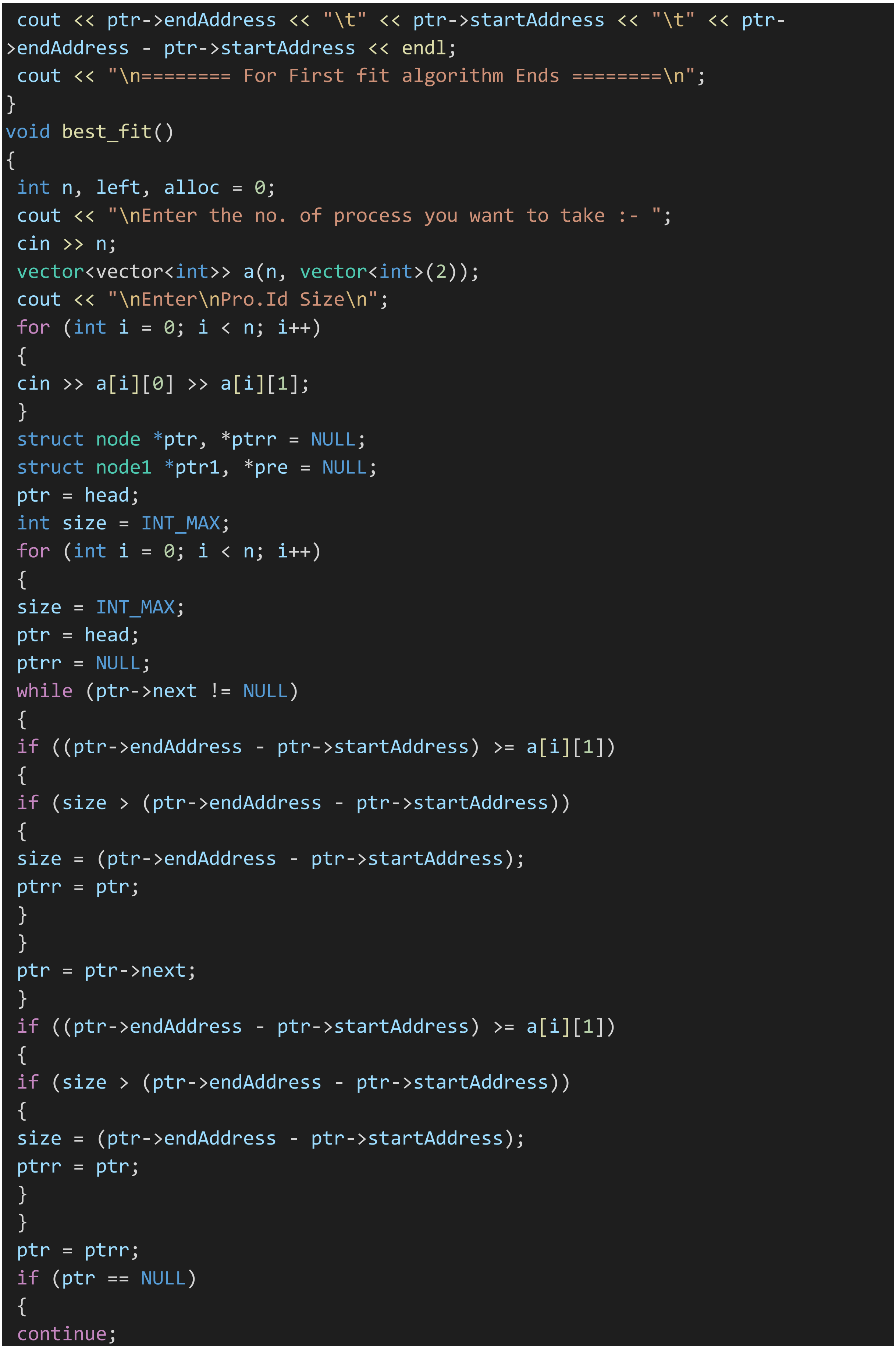


```
 pre = o;
 for (int i = 1; i < 10; i++)
 {
 p = createnode();
 pre->next = p;
 p->pre = pre;
 p->next = NULL;
 p->startAddress = pre->endAddress + i * 10;
 p->endAddress = p->startAddress + (i * 50) + i * i;
 pre = p;
 }
 p = createnode();
 p->endAddress = 3150;
 p->startAddress = 3000;
 p->next = NULL;
 p->pre = pre;
 pre->next = p;
 ptr = head;
 tail = p;
 /*
while(ptr->next!=NULL)
{
cout<<ptr->endAddress<<" - "<<ptr->startAddress<<" = "<<ptr->endAddress-ptr-
>startAddress<<endl;
ptr=ptr->next;
}
cout<<ptr->endAddress<<" - "<<ptr->startAddress<<" = "<<ptr->endAddress-ptr-
>startAddress<<endl;
*/
 return;
}
void first_fit()
{
 int n, left, alloc = 0;
 cout << "\nEnter the no. of process you want to take :- ";
 cin >> n;
 vector<vector<int> > a(n, vector<int>(2));
 cout << "\nEnter\nPro.Id Size\n";
 for (int i = 0; i < n; i++)
 {
 cin >> a[i][0] >> a[i][1];
 }
 struct node *ptr;
 struct node1 *ptr1, *pre = NULL;
 ptr = head;
 int size = 0;
 for (int i = 0; i < n; i++)
 {
```

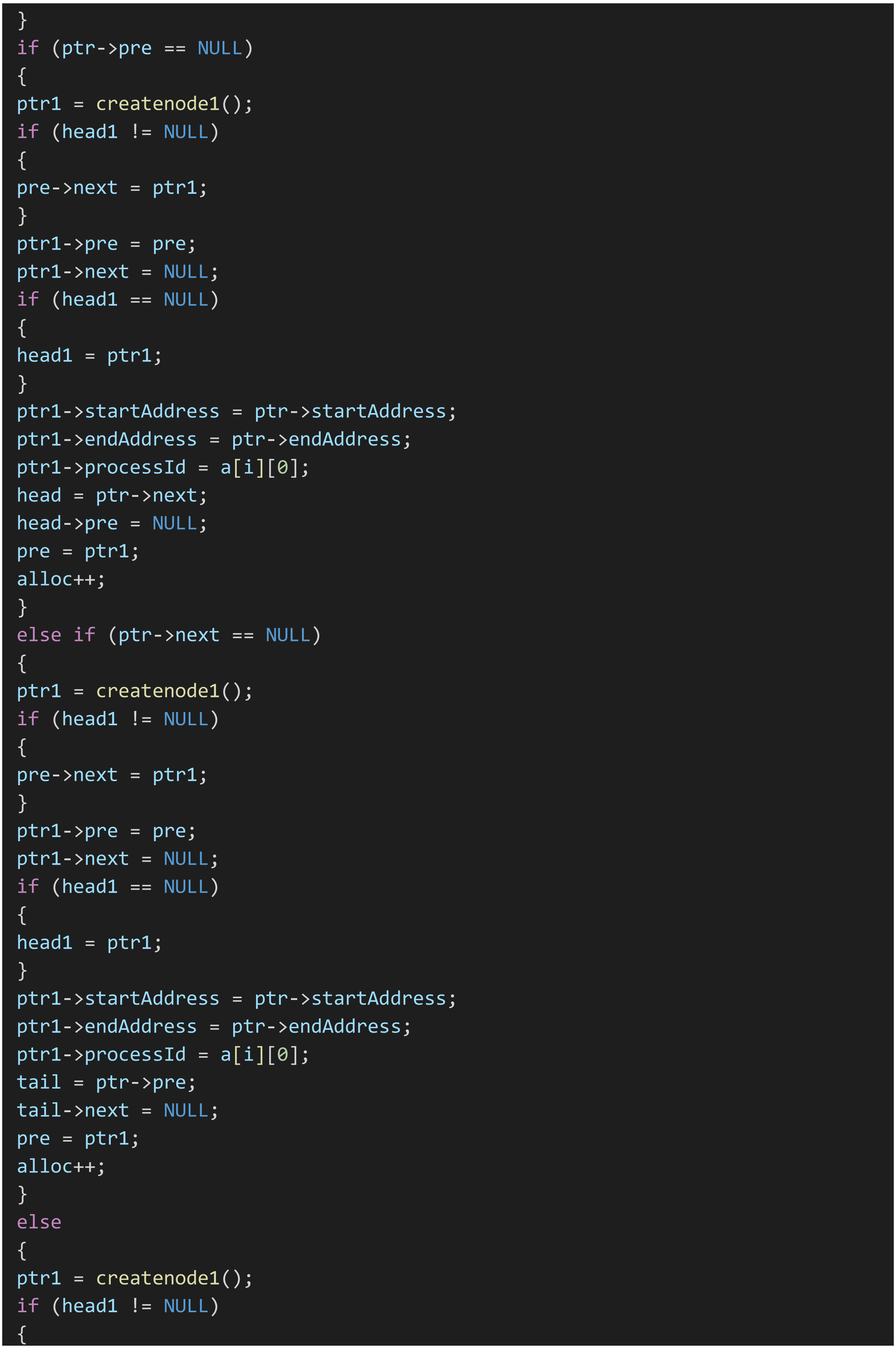


```
ptr = head;
while (ptr->next != NULL && (ptr->endAddress - ptr->startAddress) < a[i][1])
{
ptr = ptr->next;
}
if (ptr->pre == NULL)
{
ptr1 = createnode1();
if (head1 != NULL)
{
pre->next = ptr1;
}
ptr1->pre = pre;
ptr1->next = NULL;
if (head1 == NULL)
{
head1 = ptr1;
}
ptr1->startAddress = ptr->startAddress;
ptr1->endAddress = ptr->endAddress;
ptr1->processId = a[i][0];
head = ptr->next;
head->pre = NULL;
pre = ptr1;
alloc++;
}
else if (ptr->next == NULL && (ptr->endAddress - ptr-
>startAddress) >= a[i][1])
{
ptr1 = createnode1();
if (head1 != NULL)
{
pre->next = ptr1;
}
ptr1->pre = pre;
ptr1->next = NULL;
if (head1 == NULL)
{
head1 = ptr1;
}
ptr1->startAddress = ptr->startAddress;
ptr1->endAddress = ptr->endAddress;
ptr1->processId = a[i][0];
tail = ptr->pre;
tail->next = NULL;
pre = ptr1;
alloc++;
}
```

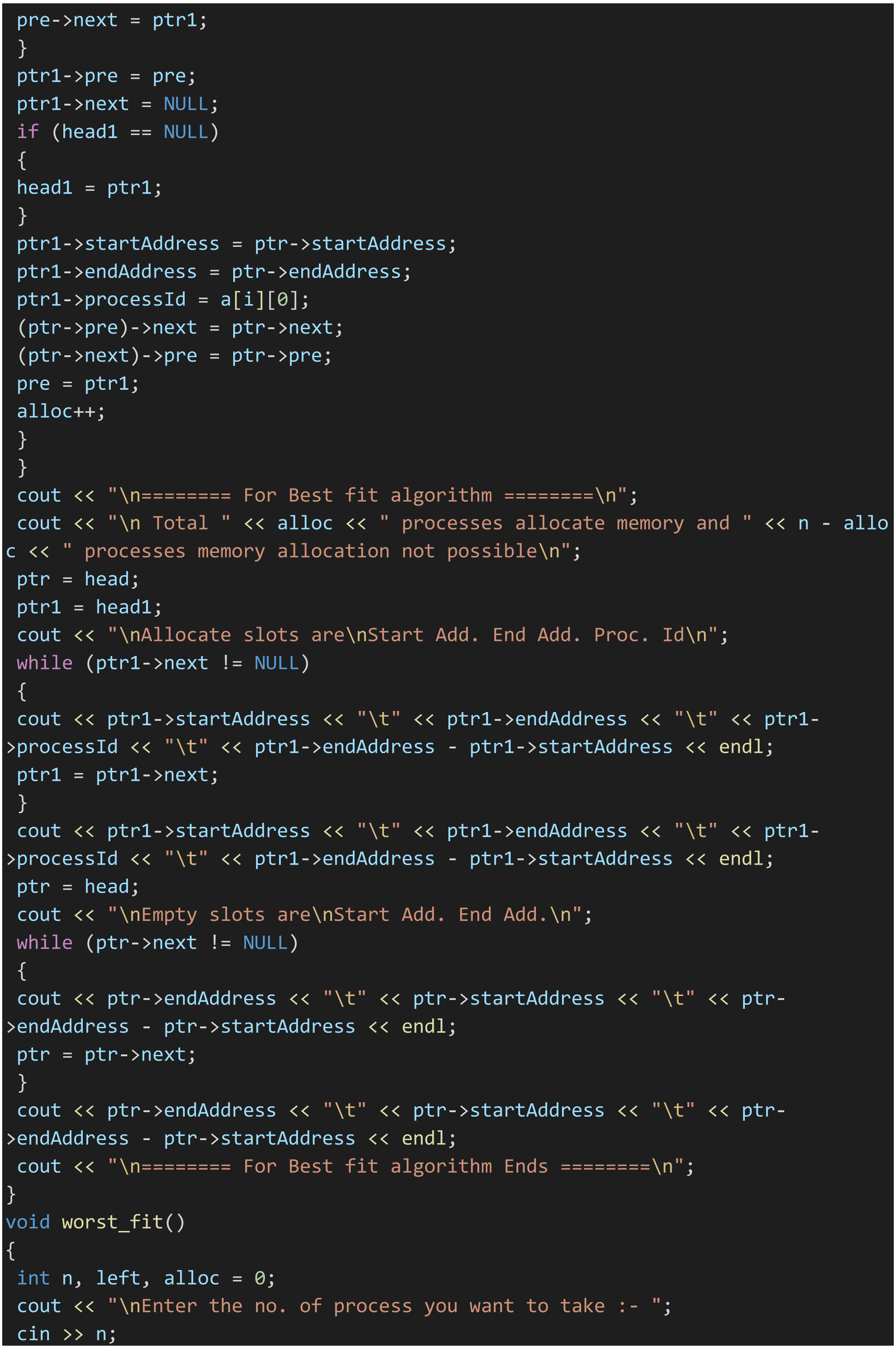
```
 else if (ptr->next == NULL)
 {
 continue;
 }
 else
 {
 ptr1 = createnode1();
 if (head1 != NULL)
 {
 pre->next = ptr1;
 }
 ptr1->pre = pre;
 ptr1->next = NULL;
 if (head1 == NULL)
 {
 head1 = ptr1;
 }
 ptr1->startAddress = ptr->startAddress;
 ptr1->endAddress = ptr->endAddress;
 ptr1->processId = a[i][0];
 (ptr->pre)->next = ptr->next;
 (ptr->next)->pre = ptr->pre;
 pre = ptr1;
 alloc++;
 }
 }
 cout << "\n======== For First fit algorithm ========\n";
 cout << "\n Total " << alloc << " processes allocate memory and " << n - allo
c << " processes memory allocation not possible\n";
 ptr = head;
 ptr1 = head1;
 cout << "\nAllocate slots are\nStart Add. End Add. Proc. Id\n";
 while (ptr1->next != NULL)
 {
 cout << ptr1->startAddress << "\t" << ptr1->endAddress << "\t" << ptr1-
>processId << "\t" << ptr1->endAddress - ptr1->startAddress << endl;
 ptr1 = ptr1->next;
 }
 cout << ptr1->startAddress << "\t" << ptr1->endAddress << "\t" << ptr1-
>processId << "\t" << ptr1->endAddress - ptr1->startAddress << endl;
 ptr = head;
 cout << "\nEmpty slots are\nStart Add. End Add.\n";
 while (ptr->next != NULL)
 {
 cout << ptr->endAddress << "\t" << ptr->startAddress << "\t" << ptr-
>endAddress - ptr->startAddress << endl;
 ptr = ptr->next;
 }
```



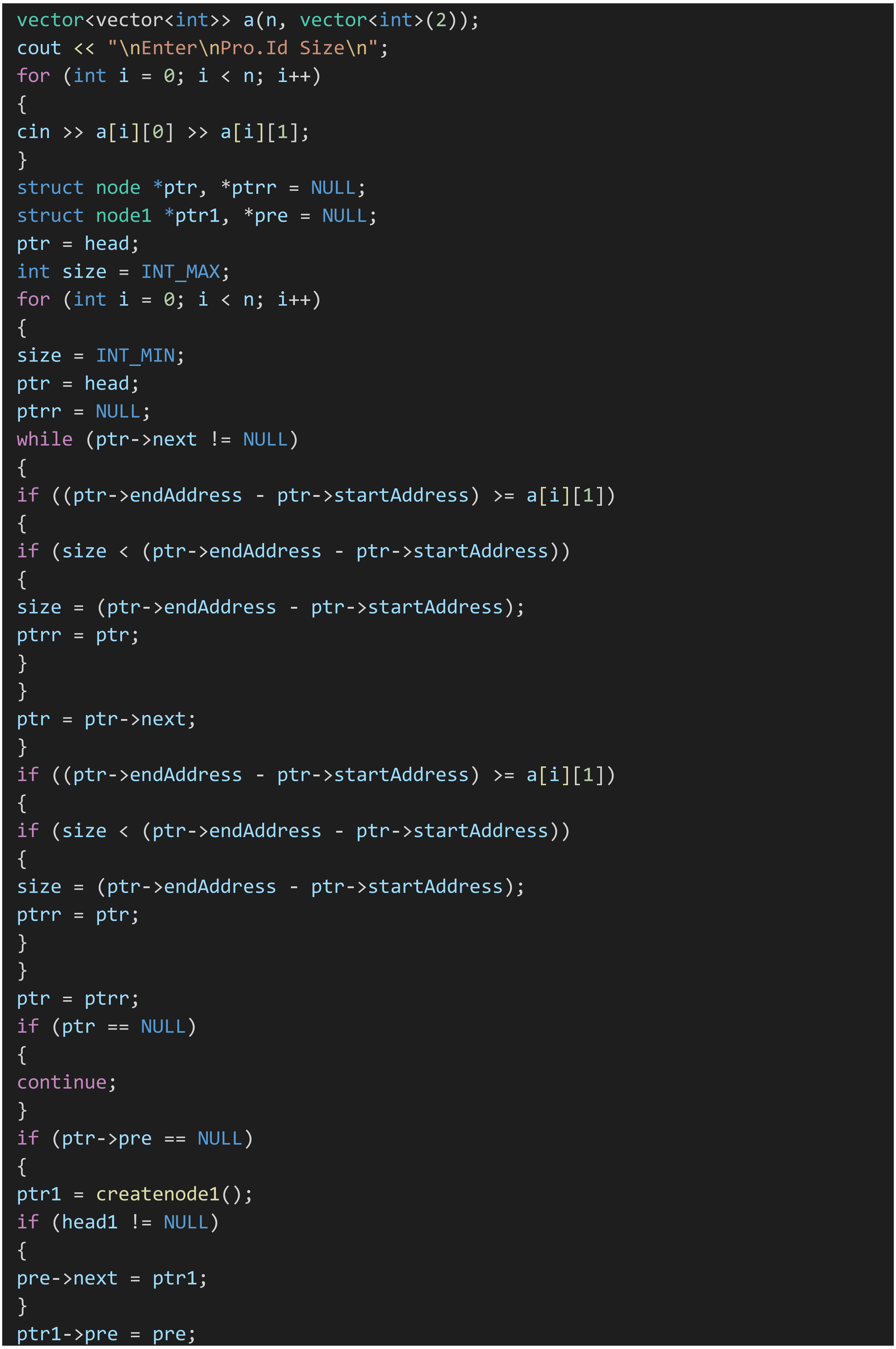
```
 cout << ptr->endAddress << "\t" << ptr->startAddress << "\t" << ptr-
>endAddress - ptr->startAddress << endl;
 cout << "\n======== For First fit algorithm Ends ========\n";
}
void best_fit()
{
 int n, left, alloc = 0;
 cout << "\nEnter the no. of process you want to take :- ";
 cin >> n;
 vector<vector<int>> a(n, vector<int>(2));
 cout << "\nEnter\nPro.Id Size\n";
 for (int i = 0; i < n; i++)
 {
 cin >> a[i][0] >> a[i][1];
 }
 struct node *ptr, *ptrr = NULL;
 struct node1 *ptr1, *pre = NULL;
 ptr = head;
 int size = INT_MAX;
 for (int i = 0; i < n; i++)
 {
 size = INT_MAX;
 ptr = head;
 ptrr = NULL;
 while (ptr->next != NULL)
 {
 if ((ptr->endAddress - ptr->startAddress) >= a[i][1])
 {
 if (size > (ptr->endAddress - ptr->startAddress))
 {
 size = (ptr->endAddress - ptr->startAddress);
 ptrr = ptr;
 }
 }
 ptr = ptr->next;
 }
 if ((ptr->endAddress - ptr->startAddress) >= a[i][1])
 {
 if (size > (ptr->endAddress - ptr->startAddress))
 {
 size = (ptr->endAddress - ptr->startAddress);
 ptrr = ptr;
 }
 }
 ptr = ptrr;
 if (ptr == NULL)
 {
 continue;
```



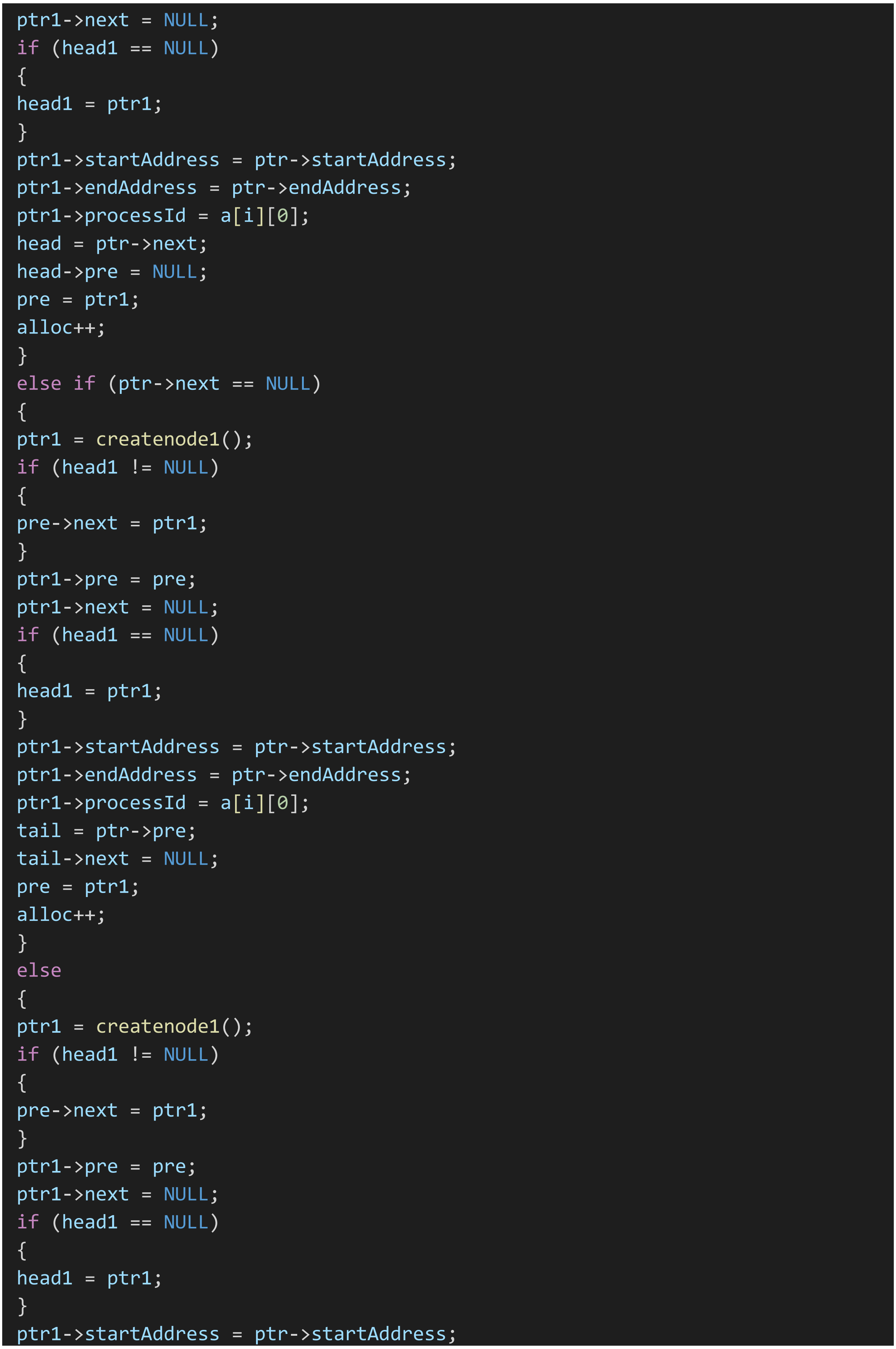
```
}
if (ptr->pre == NULL)
{
ptr1 = createnode1();
if (head1 != NULL)
{
pre->next = ptr1;
}
ptr1->pre = pre;
ptr1->next = NULL;
if (head1 == NULL)
{
head1 = ptr1;
}
ptr1->startAddress = ptr->startAddress;
ptr1->endAddress = ptr->endAddress;
ptr1->processId = a[i][0];
head = ptr->next;
head->pre = NULL;
pre = ptr1;
alloc++;
}
else if (ptr->next == NULL)
{
ptr1 = createnode1();
if (head1 != NULL)
{
pre->next = ptr1;
}
ptr1->pre = pre;
ptr1->next = NULL;
if (head1 == NULL)
{
head1 = ptr1;
}
ptr1->startAddress = ptr->startAddress;
ptr1->endAddress = ptr->endAddress;
ptr1->processId = a[i][0];
tail = ptr->pre;
tail->next = NULL;
pre = ptr1;
alloc++;
}
else
{
ptr1 = createnode1();
if (head1 != NULL)
{
```



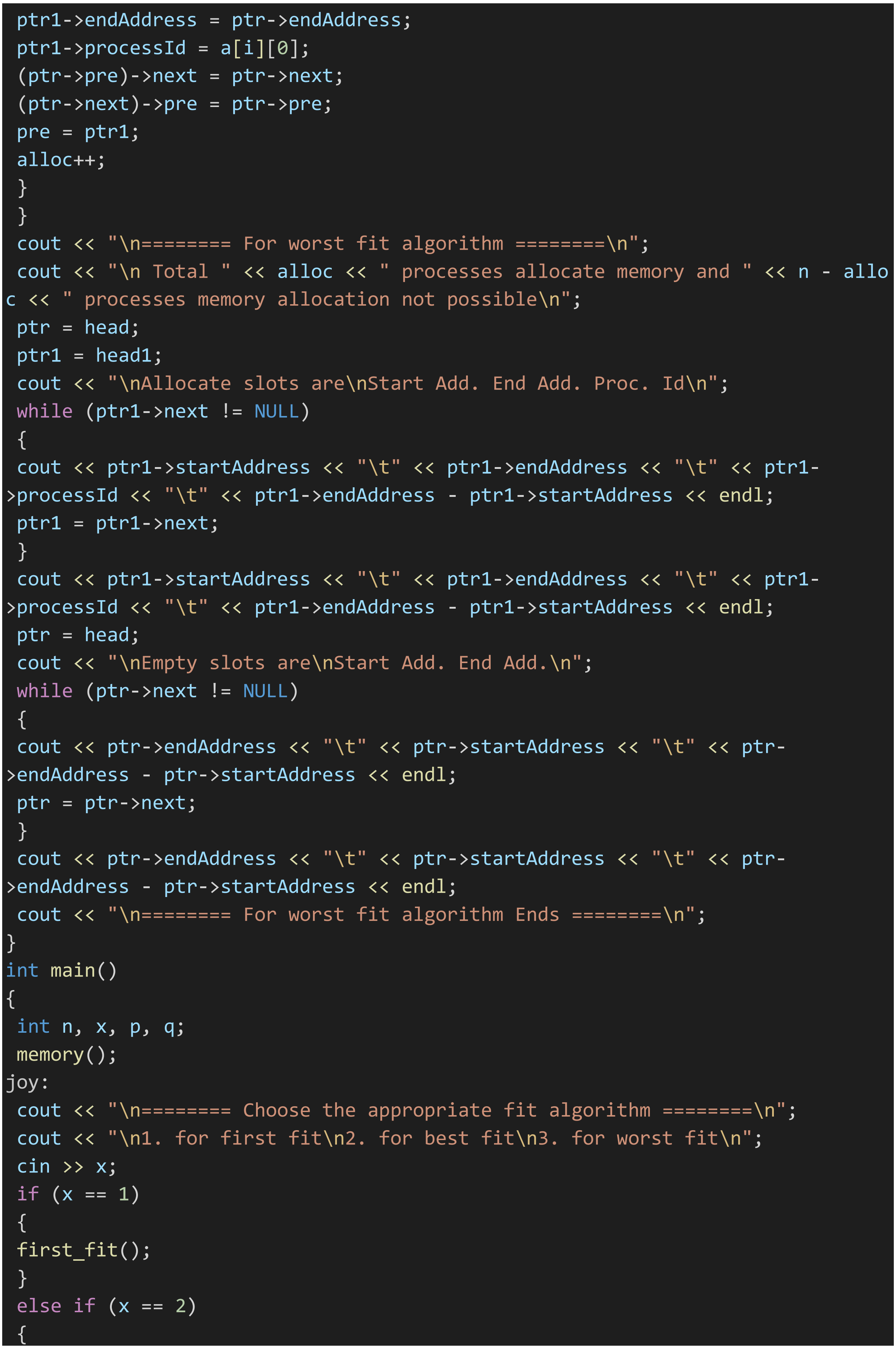
```
 pre->next = ptr1;
 }
 ptr1->pre = pre;
 ptr1->next = NULL;
 if (head1 == NULL)
 {
 head1 = ptr1;
 }
 ptr1->startAddress = ptr->startAddress;
 ptr1->endAddress = ptr->endAddress;
 ptr1->processId = a[i][0];
 (ptr->pre)->next = ptr->next;
 (ptr->next)->pre = ptr->pre;
 pre = ptr1;
 alloc++;
 }
 }
 cout << "\n======== For Best fit algorithm ========\n";
 cout << "\n Total " << alloc << " processes allocate memory and " << n - alloc << " processes memory allocation not possible\n";
 ptr = head;
 ptr1 = head1;
 cout << "\nAllocate slots are\nStart Add. End Add. Proc. Id\n";
 while (ptr1->next != NULL)
 {
 cout << ptr1->startAddress << "\t" << ptr1->endAddress << "\t" << ptr1->processId << "\t" << ptr1->endAddress - ptr1->startAddress << endl;
 ptr1 = ptr1->next;
 }
 cout << ptr1->startAddress << "\t" << ptr1->endAddress << "\t" << ptr1->processId << "\t" << ptr1->endAddress - ptr1->startAddress << endl;
 ptr = head;
 cout << "\nEmpty slots are\nStart Add. End Add.\n";
 while (ptr->next != NULL)
 {
 cout << ptr->endAddress << "\t" << ptr->startAddress << "\t" << ptr->endAddress - ptr->startAddress << endl;
 ptr = ptr->next;
 }
 cout << ptr->endAddress << "\t" << ptr->startAddress << "\t" << ptr->endAddress - ptr->startAddress << endl;
 cout << "\n======== For Best fit algorithm Ends ========\n";
}
void worst_fit()
{
 int n, left, alloc = 0;
 cout << "\nEnter the no. of process you want to take :- ";
 cin >> n;
```



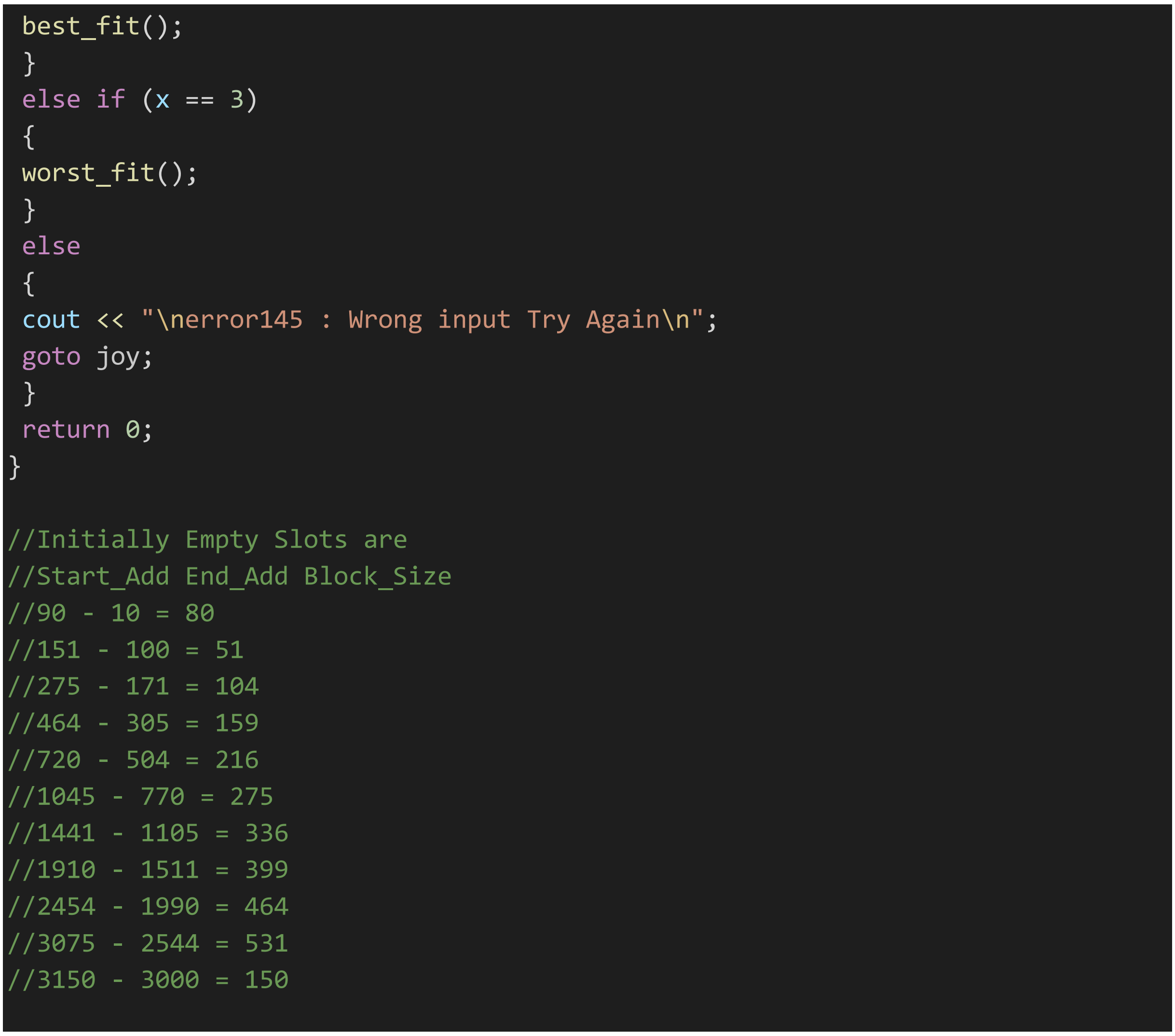
```
vector<vector<int>> a(n, vector<int>(2));
cout << "\nEnter\nPro.Id Size\n";
for (int i = 0; i < n; i++)
{
cin >> a[i][0] >> a[i][1];
}
struct node *ptr, *ptrr = NULL;
struct node1 *ptr1, *pre = NULL;
ptr = head;
int size = INT_MAX;
for (int i = 0; i < n; i++)
{
size = INT_MIN;
ptr = head;
ptrr = NULL;
while (ptr->next != NULL)
{
if ((ptr->endAddress - ptr->startAddress) >= a[i][1])
{
if (size < (ptr->endAddress - ptr->startAddress))
{
size = (ptr->endAddress - ptr->startAddress);
ptrr = ptr;
}
}
ptr = ptr->next;
}
if ((ptr->endAddress - ptr->startAddress) >= a[i][1])
{
if (size < (ptr->endAddress - ptr->startAddress))
{
size = (ptr->endAddress - ptr->startAddress);
ptrr = ptr;
}
}
ptr = ptrr;
if (ptr == NULL)
{
continue;
}
if (ptr->pre == NULL)
{
ptr1 = createnode1();
if (head1 != NULL)
{
pre->next = ptr1;
}
ptr1->pre = pre;
```



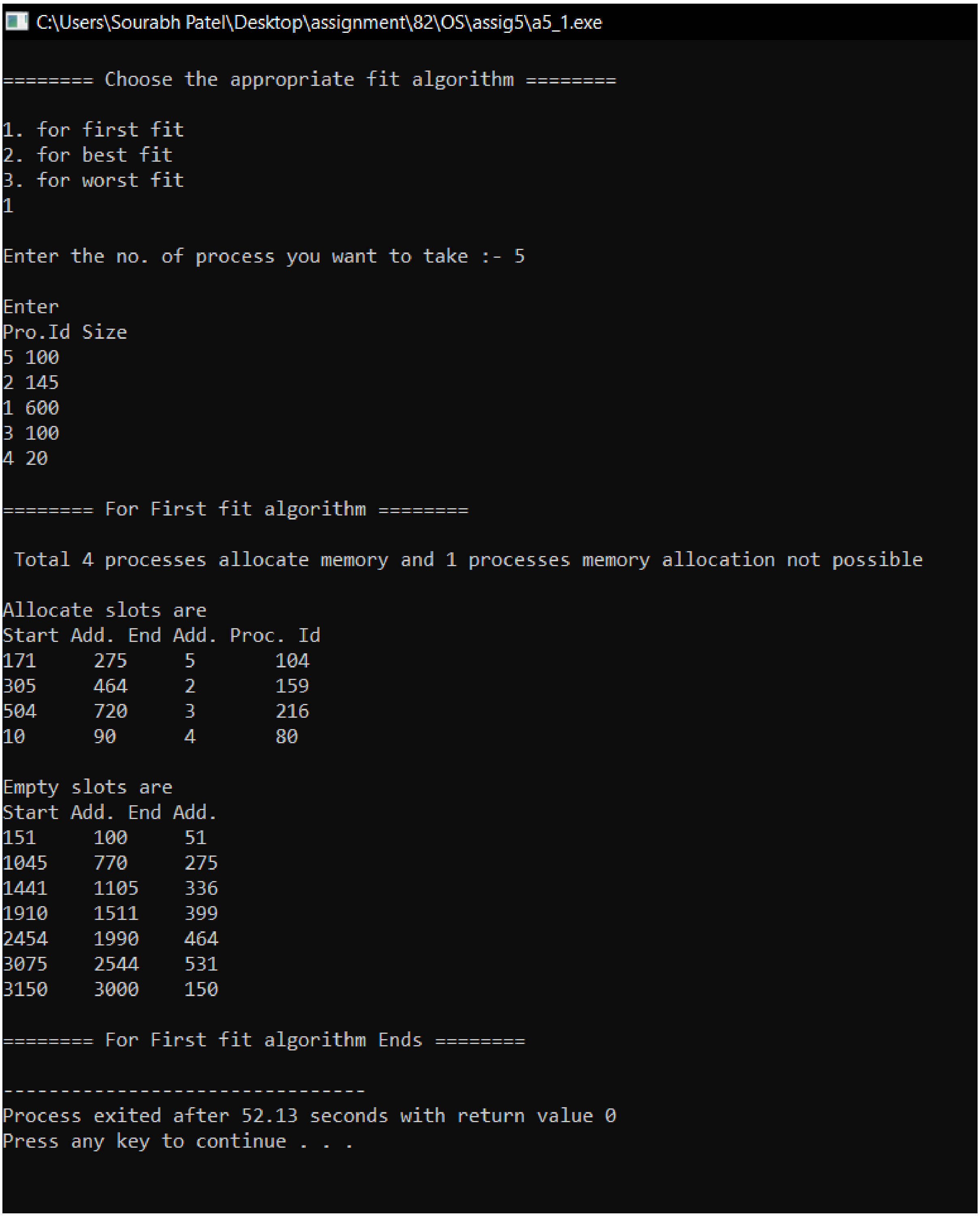
```
ptr1->next = NULL;
if (head1 == NULL)
{
head1 = ptr1;
}
ptr1->startAddress = ptr->startAddress;
ptr1->endAddress = ptr->endAddress;
ptr1->processId = a[i][0];
head = ptr->next;
head->pre = NULL;
pre = ptr1;
alloc++;
}
else if (ptr->next == NULL)
{
ptr1 = createnode1();
if (head1 != NULL)
{
pre->next = ptr1;
}
ptr1->pre = pre;
ptr1->next = NULL;
if (head1 == NULL)
{
head1 = ptr1;
}
ptr1->startAddress = ptr->startAddress;
ptr1->endAddress = ptr->endAddress;
ptr1->processId = a[i][0];
tail = ptr->pre;
tail->next = NULL;
pre = ptr1;
alloc++;
}
else
{
ptr1 = createnode1();
if (head1 != NULL)
{
pre->next = ptr1;
}
ptr1->pre = pre;
ptr1->next = NULL;
if (head1 == NULL)
{
head1 = ptr1;
}
ptr1->startAddress = ptr->startAddress;
```



```
 ptr1->endAddress = ptr->endAddress;
 ptr1->processId = a[i][0];
 (ptr->pre)->next = ptr->next;
 (ptr->next)->pre = ptr->pre;
 pre = ptr1;
 alloc++;
 }
 }
 cout << "\n======== For worst fit algorithm ========\n";
 cout << "\n Total " << alloc << " processes allocate memory and " << n - allo
c << " processes memory allocation not possible\n";
 ptr = head;
 ptr1 = head1;
 cout << "\nAllocate slots are\nStart Add. End Add. Proc. Id\n";
 while (ptr1->next != NULL)
 {
 cout << ptr1->startAddress << "\t" << ptr1->endAddress << "\t" << ptr1-
>processId << "\t" << ptr1->endAddress - ptr1->startAddress << endl;
 ptr1 = ptr1->next;
 }
 cout << ptr1->startAddress << "\t" << ptr1->endAddress << "\t" << ptr1-
>processId << "\t" << ptr1->endAddress - ptr1->startAddress << endl;
 ptr = head;
 cout << "\nEmpty slots are\nStart Add. End Add.\n";
 while (ptr->next != NULL)
 {
 cout << ptr->endAddress << "\t" << ptr->startAddress << "\t" << ptr-
>endAddress - ptr->startAddress << endl;
 ptr = ptr->next;
 }
 cout << ptr->endAddress << "\t" << ptr->startAddress << "\t" << ptr-
>endAddress - ptr->startAddress << endl;
 cout << "\n======== For worst fit algorithm Ends ========\n";
}
int main()
{
 int n, x, p, q;
 memory();
joy:
 cout << "\n======== Choose the appropriate fit algorithm ========\n";
 cout << "\n1. for first fit\n2. for best fit\n3. for worst fit\n";
 cin >> x;
 if (x == 1)
 {
 first_fit();
 }
 else if (x == 2)
 {
```

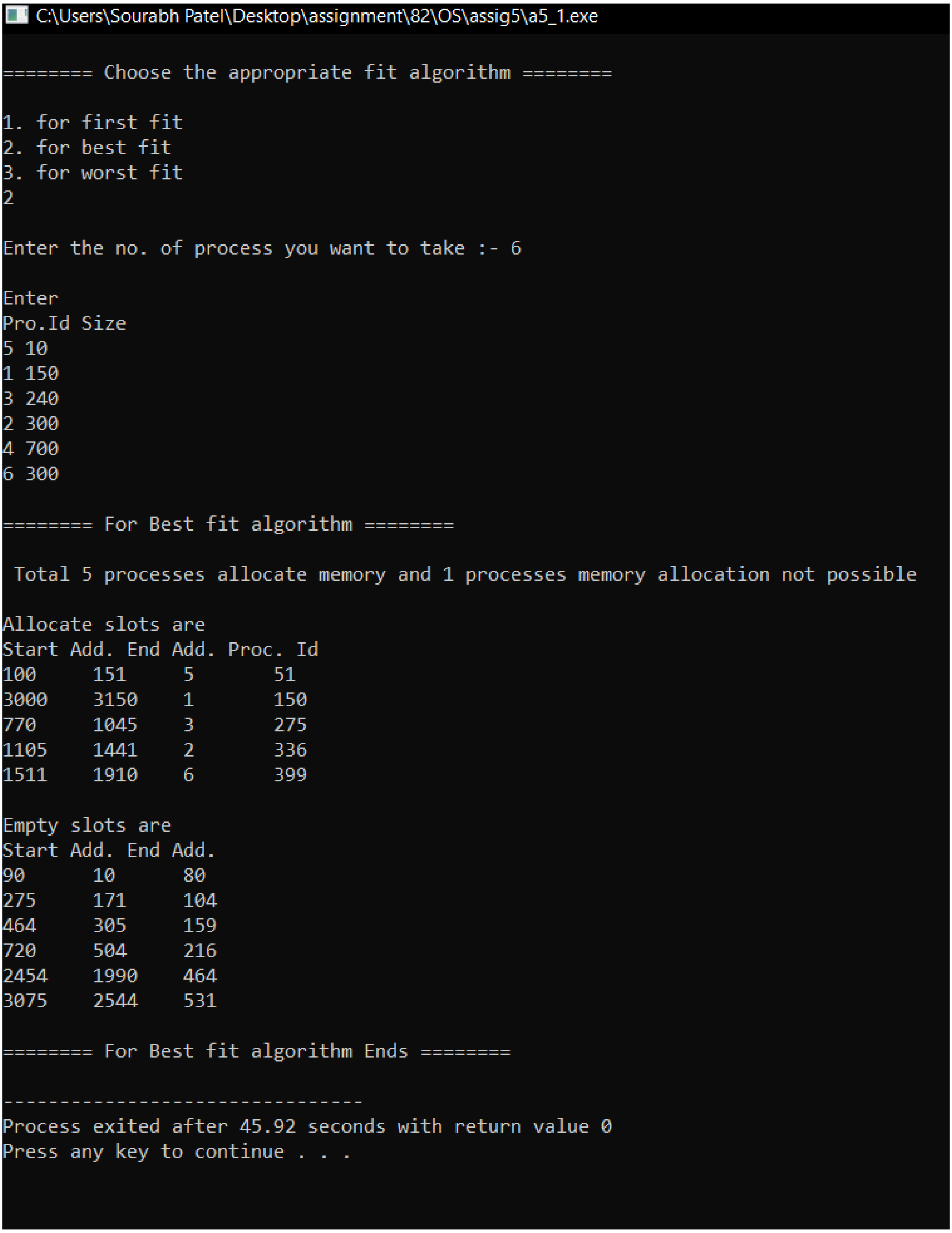


```
 best_fit();
 }
 else if (x == 3)
 {
 worst_fit();
 }
 else
 {
 cout << "\nerror145 : Wrong input Try Again\n";
 goto joy;
 }
 return 0;
}

//Initially Empty Slots are
//Start_Add End_Add Block_Size
//90 - 10 = 80
//151 - 100 = 51
//275 - 171 = 104
//464 - 305 = 159
//720 - 504 = 216
//1045 - 770 = 275
//1441 - 1105 = 336
//1910 - 1511 = 399
//2454 - 1990 = 464
//3075 - 2544 = 531
//3150 - 3000 = 150
```



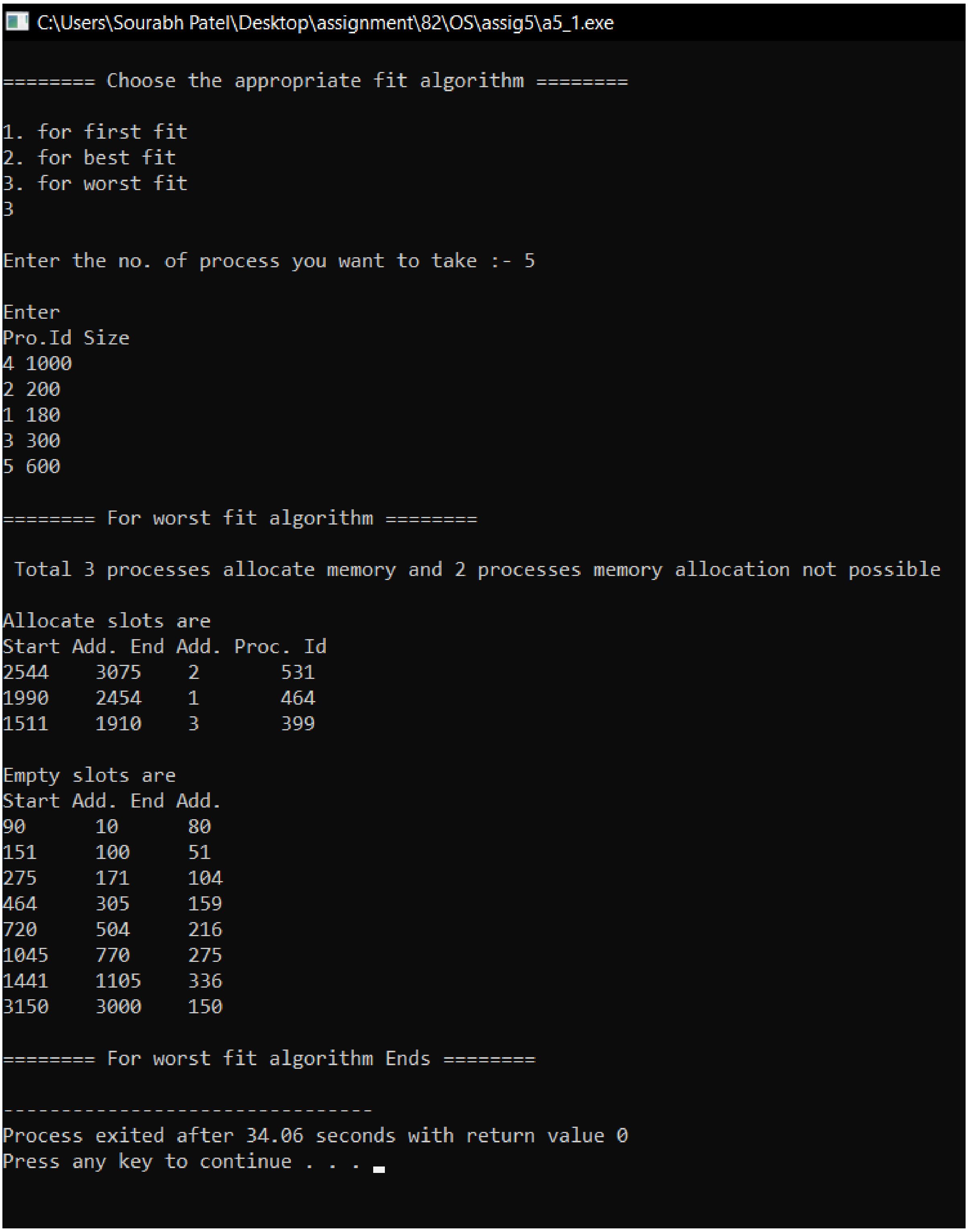
C:\Users\Sourabh Patel\Desktop\assignment\82\OS\assig5\a5_1.exe

```
======== Choose the appropriate fit algorithm ========

1. for first fit
2. for best fit
3. for worst fit
1

Enter the no. of process you want to take :- 5

Enter
Pro.Id Size
5 100
2 145
1 600
3 100
4 20

======== For First fit algorithm ========

 Total 4 processes allocate memory and 1 processes memory allocation not possible

Allocate slots are
Start Add. End Add. Proc. Id
171     275     5       104
305     464     2       159
504     720     3       216
10      90      4       80

Empty slots are
Start Add. End Add.
151     100     51
1045    770     275
1441    1105    336
1910    1511    399
2454    1990    464
3075    2544    531
3150    3000    150

======== For First fit algorithm Ends ========

--------------------------------
Process exited after 52.13 seconds with return value 0
Press any key to continue . . .
```



C:\Users\Sourabh Patel\Desktop\assignment\82\OS\assig5\a5_1.exe

```
======== Choose the appropriate fit algorithm ========

1. for first fit
2. for best fit
3. for worst fit
2

Enter the no. of process you want to take :- 6

Enter
Pro.Id Size
5 10
1 150
3 240
2 300
4 700
6 300

======== For Best fit algorithm ========

 Total 5 processes allocate memory and 1 processes memory allocation not possible

Allocate slots are
Start Add. End Add. Proc. Id
100     151     5       51
3000    3150    1       150
770     1045    3       275
1105    1441    2       336
1511    1910    6       399

Empty slots are
Start Add. End Add.
90      10      80
275     171     104
464     305     159
720     504     216
2454    1990    464
3075    2544    531

======== For Best fit algorithm Ends ========

--------------------------------
Process exited after 45.92 seconds with return value 0
Press any key to continue . . .
```

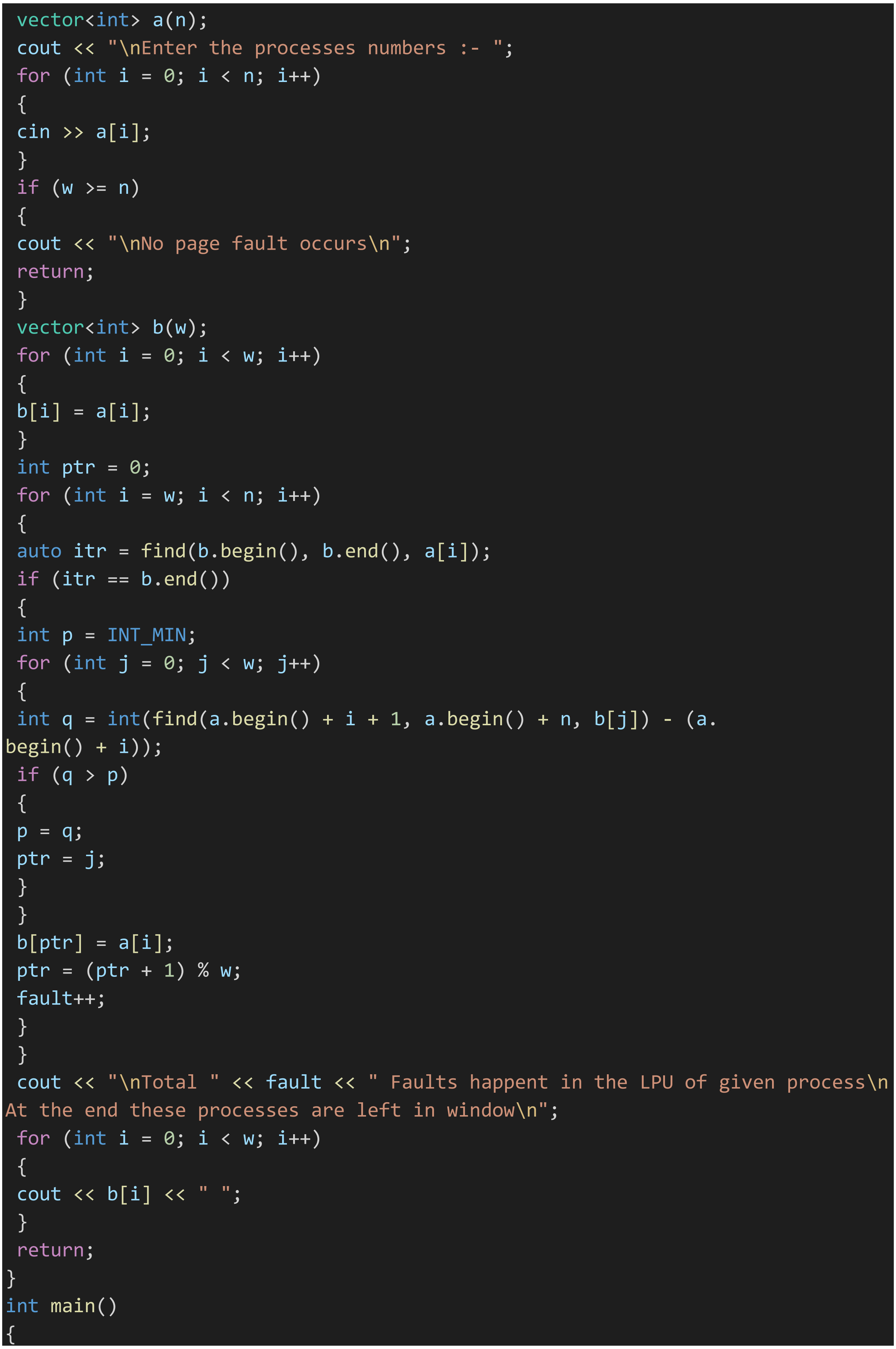


C:\Users\Sourabh Patel\Desktop\assignment\82\OS\assig5\a5_1.exe

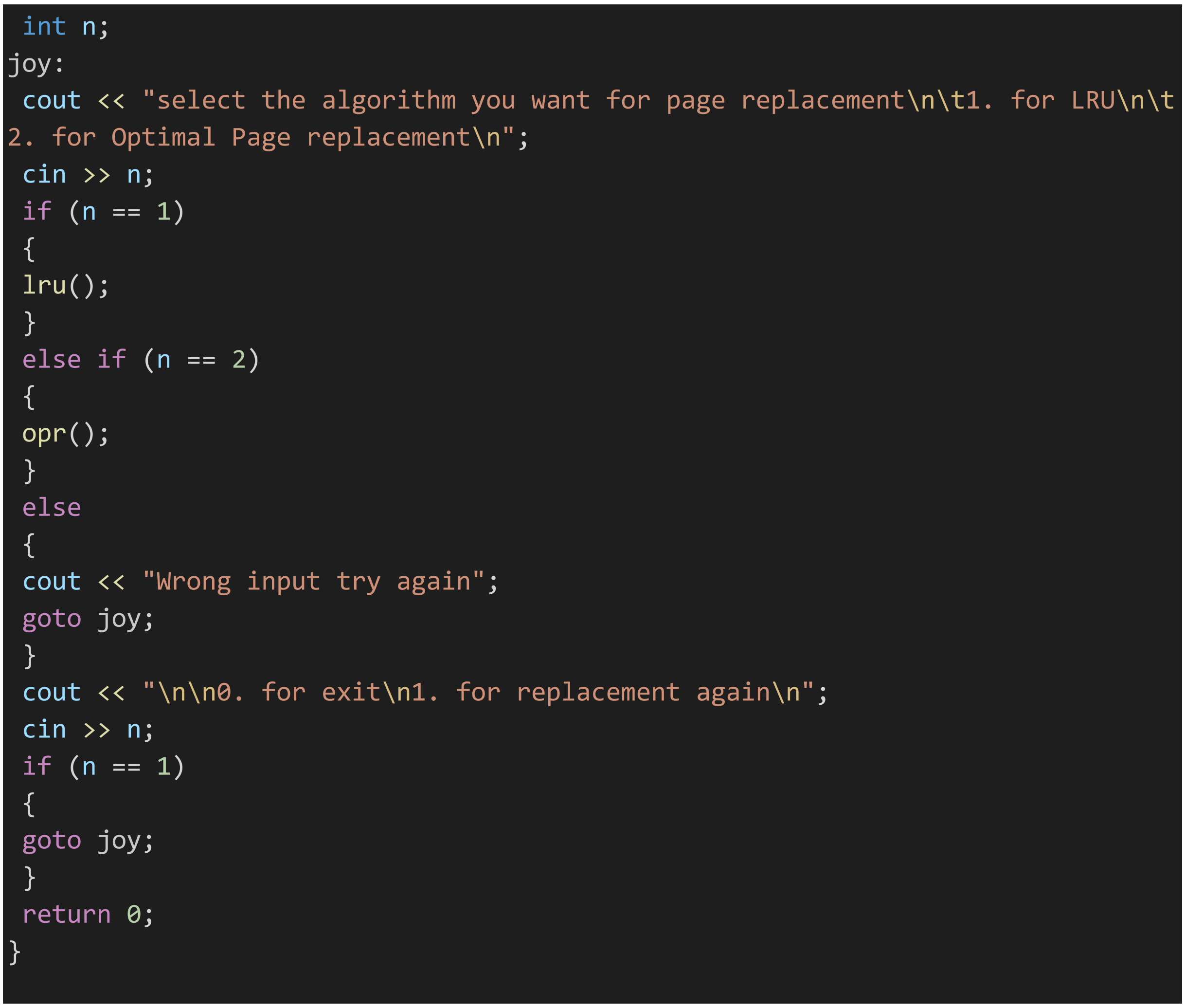
```
======== Choose the appropriate fit algorithm ========

1. for first fit
2. for best fit
3. for worst fit
3

Enter the no. of process you want to take :- 5

Enter
Pro.Id Size
4 1000
2 200
1 180
3 300
5 600

======== For worst fit algorithm ========

 Total 3 processes allocate memory and 2 processes memory allocation not possible

Allocate slots are
Start Add. End Add. Proc. Id
2544     3075     2        531
1990     2454     1        464
1511     1910     3        399

Empty slots are
Start Add. End Add.
90       10       80
151      100      51
275      171      104
464      305      159
720      504      216
1045     770      275
1441     1105     336
3150     3000     150

======== For worst fit algorithm Ends ========

--------------------------------
Process exited after 34.06 seconds with return value 0
Press any key to continue . . .
```

2. **Write a program that implements the following Page replacement algorithm. i) LRU (Least Recently Used) ii) Optimal Page Replacement algorithm .**

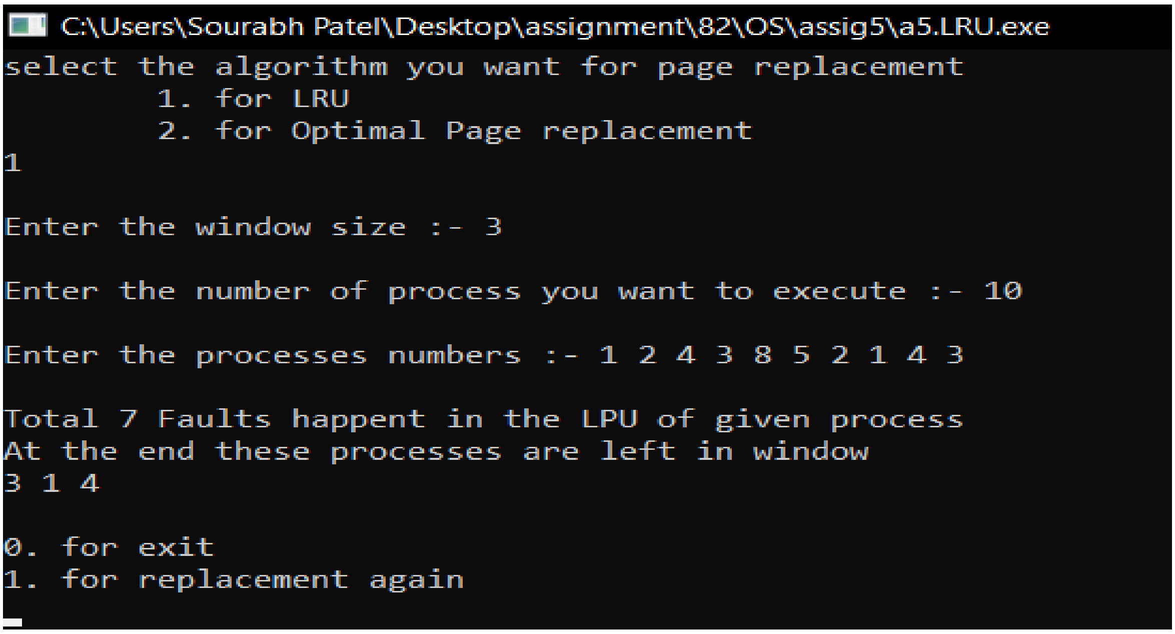
```
#include <bits/stdc++.h>
using namespace std;
void lru()
{
 int w, n, fault = 0;
 cout << "\nEnter the window size :- ";
 cin >> w;
 cout << "\nEnter the number of process you want to execute :- ";
 cin >> n;
 vector<int> a(n);
 cout << "\nEnter the processes numbers :- ";
 for (int i = 0; i < n; i++)
 {
 cin >> a[i];
 }
 if (w >= n)
 {
 cout << "\nNo page fault occurs\n";
 return;
 }
 vector<int> b(w);
 for (int i = 0; i < w; i++)
 {
 b[i] = a[i];
 }
 int ptr = 0;
 for (int i = w; i < n; i++)
 {
 auto itr = find(b.begin(), b.end(), a[i]);
 if (itr == b.end())
 {
 //cout<<b[0]<<" "<<b[1]<<" "<<b[2]<<" "<<a[i]<<" "<<fault<<endl;
 b[ptr] = a[i];
 ptr = (ptr + 1) % w;
 fault++;
 //cout<<b[0]<<" "<<b[1]<<" "<<b[2]<<" "<<a[i]<<" "<<fault<<endl<<endl;
 }
 }
 cout << "\nTotal " << fault << " Faults happent in the LPU of given process\n
At the end these processes are left in window\n";
 for (int i = 0; i < w; i++)
 {
 cout << b[i] << " ";
 }
 return;
}
void opr()
{
 int w, n, fault = 0;
 cout << "\nEnter the window size :- ";
 cin >> w;
 cout << "\nEnter the number of process you want to execute :- ";
 cin >> n;
```



```
 vector<int> a(n);
 cout << "\nEnter the processes numbers :- ";
 for (int i = 0; i < n; i++)
 {
 cin >> a[i];
 }
 if (w >= n)
 {
 cout << "\nNo page fault occurs\n";
 return;
 }
 vector<int> b(w);
 for (int i = 0; i < w; i++)
 {
 b[i] = a[i];
 }
 int ptr = 0;
 for (int i = w; i < n; i++)
 {
 auto itr = find(b.begin(), b.end(), a[i]);
 if (itr == b.end())
 {
 int p = INT_MIN;
 for (int j = 0; j < w; j++)
 {
 int q = int(find(a.begin() + i + 1, a.begin() + n, b[j]) - (a.
begin() + i));
 if (q > p)
 {
 p = q;
 ptr = j;
 }
 }
 b[ptr] = a[i];
 ptr = (ptr + 1) % w;
 fault++;
 }
 }
 cout << "\nTotal " << fault << " Faults happent in the LPU of given process\n
At the end these processes are left in window\n";
 for (int i = 0; i < w; i++)
 {
 cout << b[i] << " ";
 }
 return;
}
int main()
{
```

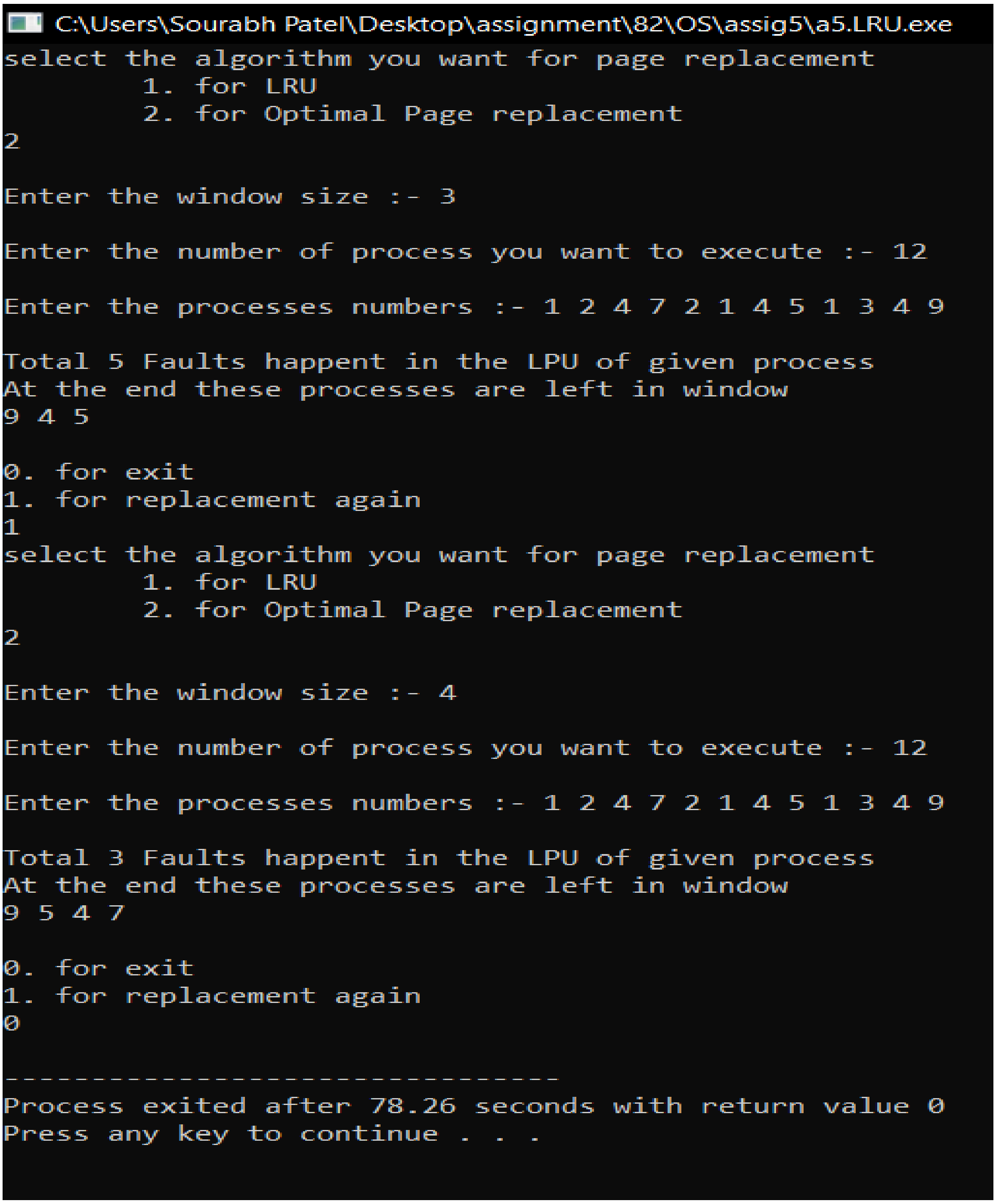


```
 int n;
joy:
 cout << "select the algorithm you want for page replacement\n\t1. for LRU\n\t
2. for Optimal Page replacement\n";
 cin >> n;
 if (n == 1)
 {
 lru();
 }
 else if (n == 2)
 {
 opr();
 }
 else
 {
 cout << "Wrong input try again";
 goto joy;
 }
 cout << "\n\n0. for exit\n1. for replacement again\n";
 cin >> n;
 if (n == 1)
 {
 goto joy;
 }
 return 0;
}
```



C:\Users\Sourabh Patel\Desktop\assignment\82\OS\assig5\a5.LRU.exe

```
select the algorithm you want for page replacement
        1. for LRU
        2. for Optimal Page replacement
1

Enter the window size :- 3

Enter the number of process you want to execute :- 10

Enter the processes numbers :- 1 2 4 3 8 5 2 1 4 3

Total 7 Faults happent in the LPU of given process
At the end these processes are left in window
3 1 4

0. for exit
1. for replacement again
```



C:\Users\Sourabh Patel\Desktop\assignment\82\OS\assig5\a5.LRU.exe

```
select the algorithm you want for page replacement
        1. for LRU
        2. for Optimal Page replacement
2

Enter the window size :- 3

Enter the number of process you want to execute :- 12

Enter the processes numbers :- 1 2 4 7 2 1 4 5 1 3 4 9

Total 5 Faults happent in the LPU of given process
At the end these processes are left in window
9 4 5

0. for exit
1. for replacement again
1
select the algorithm you want for page replacement
        1. for LRU
        2. for Optimal Page replacement
2

Enter the window size :- 4

Enter the number of process you want to execute :- 12

Enter the processes numbers :- 1 2 4 7 2 1 4 5 1 3 4 9

Total 3 Faults happent in the LPU of given process
At the end these processes are left in window
9 5 4 7

0. for exit
1. for replacement again
0

--------------------------------
Process exited after 78.26 seconds with return value 0
Press any key to continue . . .
```