/\*

GOPIKRISHNA V

S3 CSE A

52

\*/

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*n\_link;

};

struct node \*avail = NULL,\*current\_node, \*new\_node, \*prev, \*temp, \*smallest\_add;

int bal = 0, smallest = 0, best, count = 0;

int n\_blocks, n\_process, block, process;

struct node \*get\_node(int ele)

{

temp = (struct node \*)malloc(sizeof(struct node));

if(temp == NULL)

return NULL;

else

{

temp->data = ele;

temp->n\_link = NULL;

}

return temp;

}

void insert(int ele)

{

new\_node = get\_node(ele);

if(new\_node != NULL)

{

if(avail == NULL)

avail = new\_node;

else

{

current\_node = avail;

while(current\_node->n\_link != NULL)

{

current\_node = current\_node->n\_link;

}

current\_node->n\_link = new\_node;

}

}

else

{

printf("No Node Created");

}

}

void display()

{

printf("Avail List\n");

current\_node = avail;

while(current\_node != NULL)

{

printf("%d", current\_node->data);

current\_node = current\_node->n\_link;

if(current\_node != NULL)

{

printf("-->");

}

}

}

void delete(struct node \*address)

{

prev = avail;

current\_node = prev->n\_link;

while(current\_node != NULL && current\_node != address)

{

count++;

current\_node = current\_node->n\_link;

prev = prev->n\_link;

}

if(current\_node != NULL)

{

prev->n\_link = current\_node->n\_link;

free(current\_node);

}

else if(current\_node == NULL && count == 0)

{

avail = NULL;

free(current\_node);

}

else if(current\_node == NULL)

{

prev->n\_link = NULL;

free(current\_node);

}

count=0;

}

int allocate(int process)

{

current\_node = avail;

smallest = 10000;

best = 0;

while(current\_node != NULL)

{

bal = current\_node->data-process;

if(smallest > bal && bal >= 0)

{

smallest = bal;

smallest\_add = current\_node;

best = current\_node->data;

}

current\_node = current\_node->n\_link;

}

if(smallest\_add == avail)

{

avail = smallest\_add->n\_link;

free(smallest\_add);

}

else

delete(smallest\_add);

return best;

}

void main()

{

printf("\nNumber of Size Blocks = ");

scanf("%d", &n\_blocks);

LABEL:

printf("Number of Process Blocks = ");

scanf("%d", &n\_process);

int a[n\_process];

if(n\_process>n\_blocks)

{

printf("Only %d Size Blocks Available\n",n\_blocks);

goto LABEL;

}

else

{

for(int i=1;i<=n\_blocks;i++)

{

printf("Size Block %d >> ",i);

scanf("%d",&block);

insert(block);

}

}

printf("\n");

for(int i=1;i<=n\_process;i++)

{

printf("Process Block %d >> ",i);

scanf("%d",&process);

a[i]=process;

}

display();

for(int i=1;i<=n\_process;i++)

{

int best\_space=allocate(a[i]);

if(best\_space==0)

printf("\n=> Lack of space for allocating '%d'\n",a[i]);

else

printf("\n=> '%d' is allocated at '%d'\n",a[i],best\_space);

}

}

**OUTPUT**

