DATA STRUCTURE ASSIGNMENT

Submitted to:

Ms. Akshara Sasidharan

Submitted by:

Devi Gireesh

S1 MCA

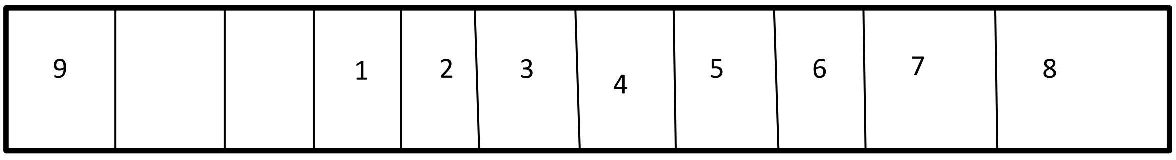
Roll No:32

1. A program P reads in 500 integers in the range[0…100] representing the scores of 500 students it then prints the frequency of each score above 50.What would be the best way for p to store the frequencies?

   We can store the frequencies of score above 50 using an array. Because 0 to 100 means the score got by 500 students. So as we represent as an array we can access the score greater than 50.

2. Consider a standard circular queue implementation (which has the same condition for queue Full and queue Empty) whose size is 11 and the elements of the queue are q[0],q[1],q[2]…q[10].The Front and rear pointers are initialized to point at q[2]. In which position will the ninth elements be added?

   Here the front and rear pointers are initialized to point at q[2]. So when we insert an new element to queue first increment REAR value with 1 that means in q[3] a new value 1 is added to it. Same way all the values upto 8 is added then the rear increment to q[0] because the space is vacant at the beginning. Then the value 9 is added to q[0].



3. Write a C program to implement red black tree.

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
char color;
```

```
struct node *left, *right, *parent;
};
void LeftRotate(struct node **root,struct node *x)
{
struct node *y = x->right;
x->right = y->left;
if (x->right != NULL)
x->right->parent = x;
y->parent = x->parent;
if (x->parent == NULL)
(*root) = y;
else if (x == x->parent->left)
x->parent->left = y;
else
x->parent->right = y;
y->left = x;
x->parent = y;
}
void rightRotate(struct node **root,struct node *y)
{
struct node *x = y->left;
y->left = x->right;
if (x->right != NULL)
x->right->parent = y;
x->parent =y->parent;
if (x->parent == NULL)
(*root) = x;
else if (y == y->parent->left)
y->parent->left = x;
else
{
y->parent->right = x;
x->right = y;
y->parent = x;
}
void insertFixUp(struct node **root,struct node *z)
{
```

```
while (z != *root && z->parent->color == 'R')
{
struct node *y;
if (z->parent == z->parent->parent->left)
y = z->parent->parent->right;
else
y = z->parent->parent->left;
if (y->color == 'R')
{
y->color = 'B';
z->parent->color = 'B';
z->parent->parent->color = 'R';
z = z->parent->parent;
}
else
{
if (z->parent == z->parent->parent->left && z == z-
>parent->left)
{
char ch = z->parent->color ;
z->parent->color = z->parent->parent->color;
z->parent->parent->color = ch;
rightRotate(root,z->parent->parent);
}
if (z->parent == z->parent->parent->left && z == z-
>parent->right)
{
char ch = z->color ;
z->color = z->parent->parent->color;
z->parent->parent->color = ch;
LeftRotate(root,z->parent);
rightRotate(root,z->parent->parent);
}
if (z->parent == z->parent->parent->right && z == z-
>parent->right)
{
char ch = z->parent->color ;
z->parent->color = z->parent->parent->color;
z->parent->parent->color = ch;
LeftRotate(root,z->parent->parent);
}
if (z->parent == z->parent->parent->right && z == z-
>parent->left)
{
char ch = z->color ;
z->color = z->parent->parent->color;
z->parent->parent->color = ch;
rightRotate(root,z->parent);
LeftRotate(root,z->parent->parent);
}
}
}
(*root)->color = 'B';
}
void insert(struct node **root, int data)
{
struct node *z = (struct node*)malloc(sizeof(struct node));
z->data = data;
z->left = z->right = z->parent = NULL;
if (*root == NULL)
{
z->color = 'B';
(*root) = z;
}
else
{
struct node *y = NULL;
struct node *x = (*root);
while (x != NULL)
{
y = x;
if (z->data < x->data)
x = x->left;
else
```

```
x = x->right;
}
z->parent = y;
if (z->data > y->data)
y->right = z;
else
y->left = z;
z->color = 'R';
insertFixUp(root,z);
}
}
void inorder(struct node *root)
{
if (root == NULL)
return;
inorder(root->left);
printf("%d ", root->data);
inorder(root->right);
}
int main()
{
struct node *root = NULL;
insert(&root,10);
insert(&root,20);
insert(&root,40);
insert(&root,30);
insert(&root,50);
insert(&root,35);
insert(&root,25);
insert(&root,37);
printf("inorder Traversal Is : ");
inorder(root);
return 0;
}
```