**Correct the code, if any problems noticed**

1. In the given queue, front and rear has been initialized with 0. Insert function of the queue has been given as:

void insert (int x)

{

if (rear+1==size)

cout<<”queue overflow”<<endl;

++rear=x;

}

**Answer:**

void insert(int x) {

if (rear + 1 == size) {

cout << "Queue overflow" << endl;

} else {

max[rear++] = x; // insert x at max[rear] and then increment rear

}

}

2. In the circular queue, front and rear has been initialized with -1. Insert function of the queue has been given as:

void insert (int x)

{

if (rear+1==size)

cout<<”queue overflow”<<endl;

rear+1=x;

}

**Answer:**

void insert(int x) {

if ((rear + 1) % size == front) {

cout << "Queue Overflow" << endl;

} else {

rear = (rear + 1) % size;

max[rear] = x;

}

}

3. In the display function, following code has been written:

void display()

{

int beg=0;

while(beg<=top)

cout<<stack[beg++]<<endl;

}

void display()

{

int t=top;

while(t>=0)

cout<<stack[t++]<<endl;

}

After correcting error from both the functions (if any), which version of them will be preferred?

**Answer:**

The first version of the `display` function starts from the bottom of the stack (index 0) and goes up to the top (index `top`). It assumes that `top` is the index of the top element in the stack.

void display() {

int beg = 0;

while (beg <= top)

cout << stack[beg++] << endl;

}

The second version of the `display` function starts from the top of the stack (index `top`) and goes down to the bottom (index 0). It assumes that `top` is the index of the top element in the stack.

void display() {

int t = top;

while (t >= 0)

cout << stack[t--] << endl;

}

Both versions of the code are correct and shows the same result. The only difference lies in the direction of iteration: one goes from bottom to top, while the other goes from top to bottom.

4. In a stack based template, pop function is written; find out the possible problem that

may occur in this code.

t pop()

{

t x=top->data;

if(top==NULL)

cout<<” Stack underflow”<<endl;

delete top; // Remove the top node

top=top->next; // move the top to the previous position

}

**Answer:**

t pop() {

if (top == NULL) {

cout << "Stack underflow" << endl;

return t();

}

t x = top->data;

Node\* nextNode = top->next;

delete top;

top = nextNode;

return x;

}